STATIC LOAD BALANCING USING NON-UNIFORM MESH PARTITIONING BASED ON RAY DENSITY PREDICTION FOR THE PARALLEL WAVEFRONT CONSTRUCTION METHOD

A Thesis

by

ABDULLAH FAHAD A ALYABES

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Richard L. Gibson, Jr
Committee Members,	Mark E. Everett
	Nancy M. Amato
Head of Department,	Rick J. Giardino

August 2014

Major Subject: Geophysics

ABSTRACT

The Wavefront Construction (WFC) method, which was developed based on ray theory, is one of the most efficient tools in seismic modeling. The main idea of this method is to propagate a wavefront represented by rays in a computational mesh that is interpolated whenever an accuracy criterion is violated. Recently, a parallel WFC was developed using the Standard Template Adaptive Parallel Library. However, due to wavefront density adaptivity, the parallel implementation exhibits inefficient performance owing to load imbalances between multiple processors. This paper applies a static load balancing approach based on a method for predicting future loads for a synthetic salt dome model, in order to improve the performance. The approach utilizes a preliminary conventional ray simulation to estimate the cost (future load) of each cell in the WFC's initial wavefront mesh. Then it applies a non-uniform mesh decomposition that results in a more efficient parallel WFC. Our implementation shows better and stable scalability in most WFC simulations. Overall, this paper contributes to understanding the behavior of wavefront mesh adaptability and predicting earth model complexities, and it serves as a guide for achieving the ultimate goal, a fully load-balanced parallel WFC.

DEDICATION

To my lovely family.

ACKNOWLEDGEMENTS

First, I want to thank my advisor, Dr. Gibson, for his guidance, advice, and support during my studies. I deeply appreciate his efforts in teaching me how to produce and communicate high-quality research. It was a great honor to be a student member of his group. I also extend my thanks to my thesis committee members Dr. Everett and Dr. Amato, and to Dr. Sparks for agreeing to serve on my defense committee. I would like to thank Saudi Aramco for sponsoring my graduate studies, and Dr. Saleh Aldossary for being a great mentor. Many thanks also go to Dr. Gibson's students and *Parasol Lab* students at Texas A&M University for helping me in this research. Finally, I would like to thank my parents for their usual support and prayers.

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

Several geophysical applications in seismology, including seismic acquisition, imaging, and interpretation, utilize ray methods (Alaei, 2012; Gjøystdal et al., 2002, 2007; Červený, 2005). The Wavefront Construction (WFC) method, which is an extension of conventional ray tracing, is one of the most efficient tools for seismic modeling (Carcione et al., 2002; Fehler and Huang, 2002). WFC methods have been implemented to simulate seismic wave propagation in stratified and girded models for both isotropic and anisotropic media (Chambers and Kendall, 2008; Chen, 2011; Gibson et al., 2005; Gibson Jr, 2000; Kaschwich, 2006; Lee, 2005; Vinje et al., 1999, 1996, 1993). The main idea of the WFC method is to propagate wavefronts represented by rays arranged in a triangular or quadrilateral computational mesh. The accuracy of the mesh is controlled by interpolating new rays whenever an accuracy criterion is violated, and it can therefore efficiently track amplitudes and multiple arrival times.

The simulation of seismic waves in complex 3D structures has made high performance computing an essential tool for seismic modeling. In the past several years, seismic modeling has utilized parallelization techniques involving both CPUs and GPUs (Bohlen, 2002; Grunberg et al., 2004; Komatitsch et al., 2010; Mohammadzaheri et al., 2013; Szostek and Leniak, 2012). Recently, the WFC method has been implemented as a parallel application using the Standard Template Adaptive Parallel Library (STAPL) (Jain, 2011). STAPL is a C++ development framework that allows users to implement parallel applications with a high level of abstraction by hiding specific details of the parallel programming (Buss et al., 2010).

Parallel WFC (pWFC) performance varies based on the input parameters and the desired output precision and is therefore affected by several factors, including source

locations, the earth model's complexity, and the simulation accuracy configurations. pWFC has been shown to lose a considerable degree of performance due to load imbalances (Jain, 2011). Load imbalances occur when there is a major difference in the work load distribution between CPUs, and this leads to a longer turnaround time. Load imbalances in pWFC arise mainly from the adaptivity of the wavefront mesh density during propagation.

pWFC has recently been implemented using uniform domain decomposition among the CPUs in the initialization phase. As a consequence, the rays on the wavefront mesh are almost evenly distributed between processes during the first simulation steps. Typically, rays are interpolated as the wavefront travels away from the source. Since ray tracing is the most expensive part of pWFC, it is better to have a load imbalance during the initial steps rather than later. In this project, we have therefore applied static load balancing by using an initial non-uniform distribution of rays based on wavefront mesh density predictions. We first developed a method to estimate the future cost (load) of each cell in the wavefront mesh using a preliminary ray simulation, and then we applied a non-uniform decomposition based on our load predictions. We used a synthetic Gulf of Mexico salt dome model to investigate the impact of load imbalances on pWFC performance. This research should help in understanding the behavior of pWFC load balancing as well as in deciding on the best strategy to achieve the ultimate goal, a load-balanced pWFC application.

This paper describes our static load balancing approach based on mesh density predictions. First, we review the WFC and pWFC algorithms and their performance. Then we propose three different approaches for predicting costs. We thoroughly evaluate each approach by applying pWFC to a salt dome model. Finally, we compare the performance of our proposed non-uniform wavefront mesh decomposition method to the original pWFC implementation.

2. BACKGROUND THEORY AND METHOD

2.1 Wavefront Construction Method

The WFC method, which was developed by applying a high-frequency approximation to the elastic wave equation (Červený, 2005), has been applied to isotropic and anisotropic media to simulate seismic wave propagation (Chambers and Kendall, 2008; Chen, 2011; Gibson et al., 2005; Gibson Jr, 2000; Lee, 2005; Vinje et al., 1999, 1996, 1993). The main idea of the WFC method is to trace ray fields rather than individual rays. A computational mesh represents a wavefront that connects adjacent rays at the same travel time. It begins as an initial sparse set of rays and interpolates new rays whenever an accuracy criterion is violated. By tracking the propagation of the wavefront, the WFC method addresses the two-points problem, which is to find a direct connection between the source and the receiver. WFC can therefore efficiently track amplitudes and multiple arrival times (Gibson et al., 2005; Vinje et al., 1996). The results can be utilized in different geophysical applications, such as seismic imaging (Gajewski et al., 2002; Kaschwich, 2006). It would also be very beneficial to use the WFC method for educational purposes by visualizing seismic waves in complex media (Gjøystdal et al., 2002).

The WFC algorithm used in this project consists of the following steps: (1) initialize ray directions (for the initial mesh), (2) trace individual rays, (3) construct a computational mesh that represents a wavefront, and (4) interpolate and coarsen the wavefront mesh as rays propagate through an earth model (Gibson et al., 2005; Jain, 2011; Lee and Gibson, 2007). The first step initializes a sparse set of rays from the source, for which there are two methods: the take-off angle method and the cubed sphere method. The latter method is more accurate and efficient (Lee and Gibson,

2007; Lee, 2005). This method initializes the direction of rays based on a focal cube surface surrounding the source. It then traces each ray using a predefined time step (the wavefront time step) based on asymptotic ray theory (Červený, 2005). Next, it constructs a quadrilateral mesh that connects adjacent rays at the same travel time, to represent the wavefront. Each corner in the mesh cells represents a point of the ray at a constant time, and each cell will represent a ray tube in the next propagation step. Rays typically diverge as the wavefront travels away from the source, which leads to inaccurate estimations of travel time and amplitude for points arbitrarily located within the cells (receivers) (Lee and Gibson, 2007). Therefore, the method interpolates new rays to maintain travel time accuracy (Figure 2.1). New rays are added using a specific threshold that defines the accuracy of the results based on paraxial time prediction (Gibson et al., 2005). Since ray tracing is the most expensive part of the algorithm, the method removes rays whenever the ray field reaches a sufficient density. Therefore, the density of the rays in each wavefront is unevenly distributed, based on the ray field's local behavior. During wave propagation, when a ray tube passes through a receiver on the surface, its travel time and amplitude are recorded. The main steps of the algorithm are repeated for each wavefront during the simulation (Algorithm 1). Numerical implementation details for WFC are provided in Appendix A.

2.2 Parallel Wavefront Construction Method

Geophysical applications have taken great advantage of high performance computing. Researchers have implemented different seismic modeling methods using a variety of parallel libraries and frameworks, such as the finite difference method using MPI (Bohlen, 2002), seismic ray tracing in adaptive mesh models using MPI (Grunberg et al., 2004), the high-order finite element technique using CUDA (Ko-

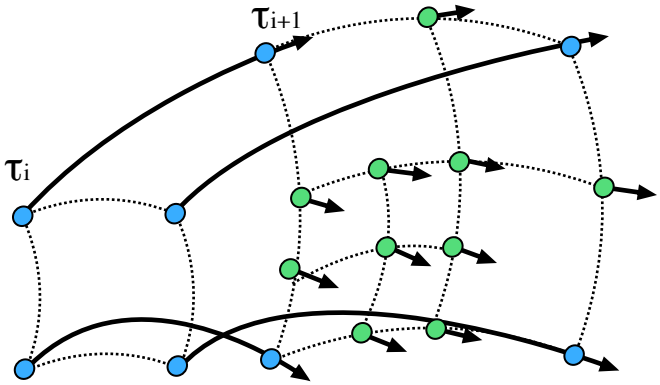


Figure 2.1: Schematic illustration of ray interpolations for one cell in the wavefront mesh from an arbitrary time τ_i to the next time step τ_{i+1}. Arrows represent rays, dotted lines represent the mesh geometry, blue circles indicate old rays, and green circles indicate new interpolated rays. The example shows more interpolated rays in the lower left part of the mesh at τ_{i+1} that have been added due to a lower degree of accuracy.

Algorithm 1 Sequential WFC Algorithm

Input: Earth model description, wavefront mesh description, source locations and receiver locations

```
1:  Initialize ray directions                          ▷ initial mesh
2:  while true do
3:      Trace individual rays by one wavefront time step
4:      Construct a wavefront quadrilateral mesh
5:      Interpolate/remove rays in/from the mesh       ▷ for next simulation step
6:      if no ray tubes remain then
7:          break
8:      end if
9:  end while
10: if a ray tube passed through a receiver on surface then
11:     Record travel time and amplitude
12: end if
```

Output: Ray path for each ray, wavefront, and arrivals (travel time and amplitude) for each receiver

matitsch et al., 2010), the WFC method based on Lomax's waveray approximation using C# threading (Szostek and Leniak, 2012), and distributed ray tracing using a map-reduce technique (Mohammadzaheri et al., 2013). Even though the WFC method is considered one of the fastest methods in seismic modeling, it is still computationally expensive to capture sufficient information about seismic waves in 3D complex media. Therefore, the parallel WFC method (pWFC) was developed using the Standard Template Adaptive Parallel Library (STAPL) (Jain, 2011).

STAPL is a C++ development framework that allows users to implement parallel applications with a high level of abstraction by hiding specific details of the parallel programming (An et al., 2003; Buss et al., 2010). It was developed to address parallel programming difficulties and to support portable parallel performance on both shared and distributed memories. STAPL was designed to play a role in parallel development similar to the role of the C++ Standard Template Library (STL) for sequential development. STL is a collection of basic algorithms (e.g., find, merge, copy, sort), generic data structures called containers (e.g., lists, sets, vectors, maps), and iterators to facilitate access to containers (Musser et al., 2001). Similar to STL, STAPL provides distributed data structures (pContainers), parallel algorithms (pAlorithms), task-dependent graphs (pRange), and pViews to facilitate data access in pContainers. More advanced users can build their own STAPL algorithms and containers and have access to runtime system components such as STAPL thread scheduling, memory management, and synchronization (Figure 2.2).

pWFC utilizes two pContainers: pMap for ray collection and pGraph for wavefronts, two pViews: pMap pView and pGraph pView, and two pAlorithms: map and map reduce (Jain, 2011). The implementation of pWFC (Algorithm 2) parallelizes the most expansive operations. It begins by partitioning the initial wavefront mesh into equal partitions for multiple processors, where rays located on the mesh parti-

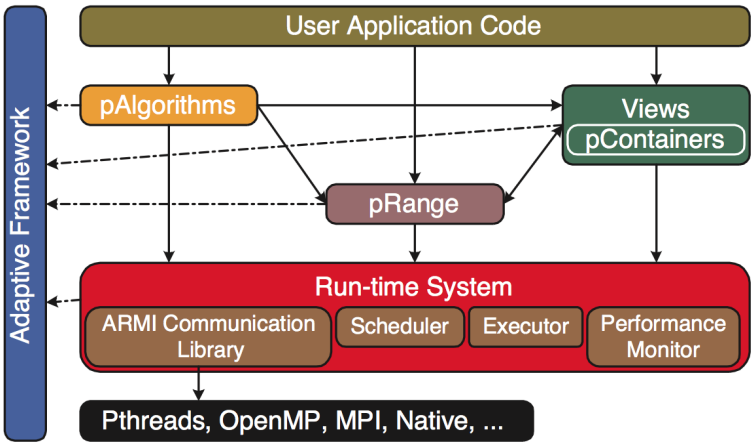


Figure 2.2: STAPL main components and their dependencies. Application developers (top level abstraction) can use pAlgorithms, pContainers, pViews, and pRange. Advanced users can access additional features (Fidel et al., 2014).

tion boundaries will be shared between processors (initialization phase; Figure 2.3a). Then, for every wavefront time step, each processor traces the rays belonging to its own partial mesh, and a partial wavefront is constructed for interpolating or coarsening rays (propagation phase). After the simulation, pWFC records multiple arrivals for each receiver on the surface (surface mapping phase).

2.3 Parallel Wavefront Construction Performance

The pWFC algorithm consists of three main phases: the initialization phase, the propagation phase, and the surface mapping phase. The main factor affecting the initialization phase's performance is the wavefront mesh description (initial number of rays and their directions). The initialization phase has been shown to have strong scalable performance using large initial meshes (large numbers of rays) (Jain, 2011). The main idea of the WFC method is to begin with a sparse set of rays, and the time taken for the initialization phase is negligible compared to the propagation phase, which is the most expensive part of the algorithm. Several input factors

Algorithm 2 Parallel WFC Algorithm

Input: Earth model description, wavefront mesh description, source locations and receiver locations

```
1: for each CPU do                                          ▷ Initialization phase
2:     Initialize ray directions                            ▷ partial initial mesh (uniform)
3: end for
4: while true do                                            ▷ Propagation phase
5:     for each CPU do
6:         Trace individual rays by one wavefront time step
7:     end for
8:     for each CPU do
9:         Construct a wavefront quadrilateral mesh
10:        Interpolate/remove rays in/from the mesh         ▷ for next simulation step
11:    end for
12:    if no ray tubes remain then
13:        break
14:    end if
15: end while
16: for each CPU do                                         ▷ Surface mapping phase
17:    if a ray tube passed through a receiver on surface then
18:        record travel time and amplitude
19:    end if
20: end for
```

Output: Ray path for each ray, wavefront, arrivals (travel time and amplitude) for each receiver, and surface map (wavefront mesh cells on the surface)

(specified by the user) can affect pWFC's propagation phase performance, such as the earth model description (isotropic, anisotropic, number of surfaces, and the choice of stratified or girded regions), the desired ray behavior (transmitted rays, reflected rays, or both), the number of sources and their locations, and the desired wave field accuracy (determined by the ray tracing step size, wavefront step size, interpolation threshold, and coarsening threshold). The propagation phase has been shown to have good scalable performance. However, load imbalances between CPUs cause a considerable degree of performance loss (Jain, 2011). Finally, two factors can affect the performance of the surface mapping phase: the number of receivers on the surface and the existence of multiple arrivals.

Load imbalances arise whenever there are large differences in the work distribution, which lead to overloaded and underloaded processors. Load imbalances are one of the most investigated issues in parallel computing, including seismic modeling (Grunberg et al., 2004). Load balancing minimizes processes' idle time through approximate uniform work distribution among the processors. There are two main categories of load balancing algorithms: static load balancing and dynamic load balancing. Static load balancing algorithms distribute work based on fixed and preconfigured rules and is usually done before execution. In contrast, dynamic load balancing algorithms are based on monitoring and redistributing the work load during execution (Hanxleden and Scott, 1991). Recently, more scalable performance of motion planning applications implemented with STAPL has been achieved using dynamic load balancing techniques (Fidel et al., 2014).

The original pWFC is implemented using initial uniform domain decomposition. In the initialization phase, the wavefront mesh is uniformly partitioned between processors (Figure 2.3a). During the first wavefront steps, rays are almost evenly distributed between processes. However, several more simulation steps may result in

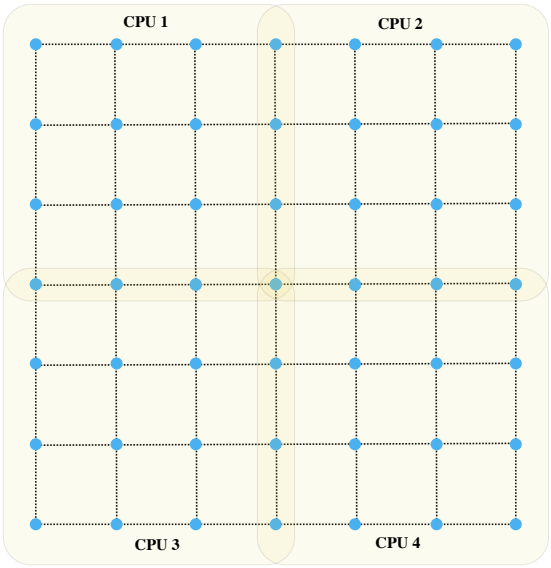
non-uniform distributions of the rays among processes. Two main factors affect ray distribution in a wavefront mesh: ray terminations and ray interpolations. Rays can be terminated for several reasons, such as when they reach the earth model boundary or hit a surface with post-critical incidence, or when the mesh is coarsened due to a high degree of accuracy. Conversely, rays can be interpolated (added) to parts of the mesh based on local ray field behavior (Figure 2.1). This adaptivity of the wavefront mesh density is the main reason for pWFC load imbalances (Figure 2.3b).

Our ultimate goal is to have load-balanced pWFC applications. To achieve this, load balancing algorithms and options must be investigated. Two dynamic load balancing algorithms have been suggested in the literature (Jain, 2011). However, in this project, we develop and investigate a new static load balancing approach that predicts the future wavefront mesh density and then non-uniformly partitions the initial wavefront mesh based on this prediction, so that the eventual partitioning will be more balanced.

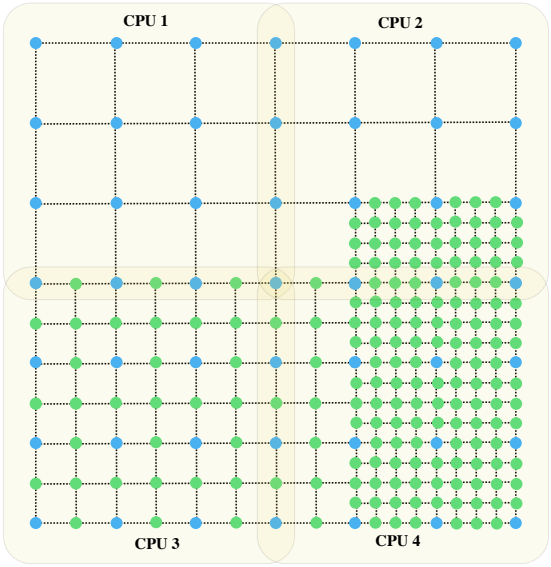
2.3.1 Wavefront Mesh Density Prediction

As the wavefront propagates through the model, new rays are interpolated into or deleted from the mesh. This behavior causes variations in the wavefront mesh density that will typically lead to non-uniform distributions of rays or ray tubes among the processors (i.e., load imbalances). Therefore, predicting the future mesh density should help in choosing the correct number of processors and their load distribution to minimize the turnaround time for pWFC. Since selecting an adequate sparse set of initial rays is very beneficial for WFC (Coman and Gajewski, 2001), this prediction should guide the choice of a sufficient density of initial rays and their directions.

We estimate the computational cost of each cell (ray tube) in the initial wavefront mesh using a preliminary ray simulation in the targeted earth model. Prior the



(a)



(b)

Figure 2.3: Schematic illustration of uniform partitioning of an initial wavefront mesh with 49 rays (36 ray tubes). Blue circles represent old rays, and green circles represent new interpolated rays. (a) Initial wavefront mesh at τ_0 that shows identical loads on all four CPUs. (b) Wavefront mesh at an arbitrary time τ_i that shows load imbalances among the CPUs.

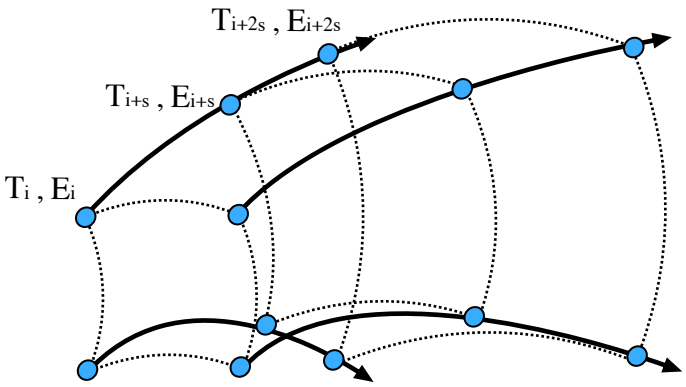


Figure 2.4: Schematic illustration of preliminary ray tracing (without interpolation) used for cost estimation. Arrows represent rays, dotted lines represent the mesh geometry, and the blue circles are rays at constant time steps. T_i and E_i are the ray tube's travel time and error at arbitrary time i. The time T and error E are recorded at each predefined time step (wavefront time step) s until the ray tube termination.

propagation phase for pWFC, we trace each ray tube in the initial wavefront mesh without doing any interpolation or coarsening. During this preliminarily simulation, we capture each ray tube's travel time and error in consecutive predefined time steps (wavefront time step; Figure 2.4). We then calculate a weight for each ray tube. We investigated three approaches to calculating the weight in this study, using (1) maximum error (Max_E), (2) maximum travel time (Max_T), and (3) maximum error and travel time (Max_ET). Finally, based on the chosen approach, we predict the computational cost of each cell based on the calculated weights.

As waves propagate away from the source, the wavefront curvature can change. The changes cause new rays to be interpolated in the WFC algorithm based on an accuracy criterion. Thus, it is reasonable to use this accuracy criterion to estimate how many rays might be interpolated in the actual WFC simulation. We have

therefore defined ray tube's error in our method as the difference between the ray tube's travel time for WFC and the predicted paraxial travel time (Appendix A.2); which is the criterion used for ray interpolation in WFC.

In our first approach to weight prediction (Max_E), we calculate the weight of a ray tube by capturing its maximum error as follows:

$$W_n = \max_{0 \leq i \leq n_{wavefronts}} E_i \tag{2.1}$$

where W_n is the weight of ray tube n, i is the time step index for the ray tube, $n_{wavefronts}$ is the number of wavefronts (time steps) of ray tube n, and E_i is the error of ray tube n at index i.

Since some ray tubes can terminate sooner than other ray tubes, we introduced travel times in our second approach to weight prediction (Max_T). In this approach, we calculate the weight by capturing the maximum travel time of each ray tube as follows:

$$W_n = \max_{0 \leq i \leq n_{wavefronts}} T_i \tag{2.2}$$

where T_i is ray tube n's travel time at time step i.

The previous two approaches to calculating weights consider the maximum time or the maximum error, which for the most ray tubes occur at the last time step. However, our third approach (Max_ET) captures both maximum travel time and maximum error at each time step. We then estimate weights by multiplying these values as follows:

$$W_n = (\max_{0 \leq i \leq n_{wavefronts}} E_i) * (\max_{0 \leq i \leq n_{wavefronts}} T_i) \tag{2.3}$$

Finally, we calculate the total computational cost of each cell in the initial wavefront mesh using one of these three approaches to estimating weight, as follows:

$$C_n = \frac{W_n}{\sum_{r=0}^{r=n_{tubes}} W_r} * 100 \tag{2.4}$$

$$\sum_{r=0}^{r=n_{tubes}} C_r = 100\% \tag{2.5}$$

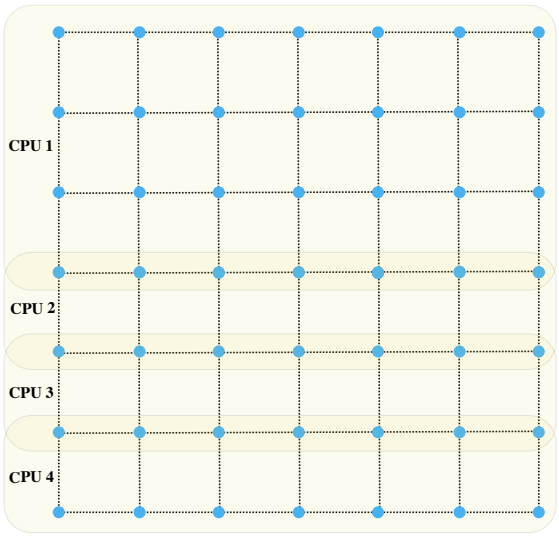
where C_n is the final cost of ray tube n, r is the the ray tube index number, n_{tubes} is the number of ray tubes (cells) in the initial mesh, W_r is the weight of ray tube r, and C_r is the final cost of ray tube r.

2.3.2 Non-uniform Wavefront Mesh Partitioning

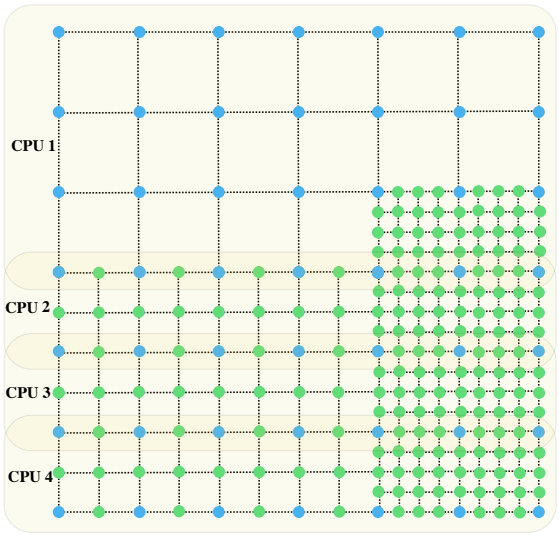
In this project we used the cubed sphere ray initialization method, which is based on the focal cube surface surrounding the source. The wavefront mesh is virtually divided into six faces: +X, -X, +Y, -Y, +Z, and -Z. These faces help in identifying the direction of rays; however, during mesh decomposition, the entire wavefront mesh is considered as a single entity.

As stated earlier, load imbalances have been observed in the uniform domain (initial mesh) decomposition implementation. However, since the number of mesh rays is much smaller during the first simulation steps, we believe it is more efficient to have a load imbalance at this stage rather than during later simulation steps. Therefore, we implemented a simple approach to non-uniformly partition the initial mesh. In our approach, we distributed the domain based on partition computational costs rather than partition sizes, by decomposing the mesh into 2D rectangular blocks. We used this approach to prove that our cost prediction is effective. Theoretical assumptions of even and uneven cell distributions are illustrated in Figures 2.3 and 2.5.

In our pWFC implementation, we introduced a new phase called the *cost prediction phase*. The pseudocode for the new pWFC is given in Algorithm 3.



(a)



(b)

Figure 2.5: Schematic illustration of non-uniform partitioning of an initial wavefront mesh with 49 rays (36 ray tubes). Blue circles represent old rays and green circles represent new interpolated rays. (a) Initial wavefront mesh at τ_0 showing the different loads on the CPUs. (b) Wavefront mesh at an arbitrary time τ_i showing a load balance among the CPUs.

Algorithm 3 New Parallel WFC Algorithm

Input: Earth model description, wavefront mesh description, source locations and receiver locations

```
1: while true do                                          ▷ Cost prediction phase
2:     for each CPU do
3:         Trace individual rays by one wavefront time step
4:     end for
5:     for each CPU do
6:         Construct a wavefront quadrilateral mesh
7:         Record travel time and error of each ray tube
8:     end for
9:     if no ray tubes remain then
10:        break
11:    end if
12: end while
13: for each ray tube do
14:    Calculate the estimated cost
15: end for
16: for each CPU do                                       ▷ Initialization phase
17:    Initialize ray directions                  ▷ non-uniform decomposition
18: end for
19: while true do                                         ▷ Propagation phase
20:    for each CPU do
21:        Trace individual rays by one wavefront time step
22:    end for
23:    for each CPU do
24:        Construct a wavefront quadrilateral mesh
25:        Interpolate/remove rays in/from the mesh      ▷ for next simulation step
26:    end for
27:    if no ray tubes remain then
28:        break
29:    end if
30: end while
31: for each CPU do                                       ▷ Surface mapping phase
32:    if a ray tube passed through a receiver on surface then
33:        record travel time and amplitude
34:    end if
35: end for
```

Output: Ray path for each ray, wavefront, arrivals (travel time and amplitude) for each receiver, and surface map (wavefront mesh cells on the surface)

3. LOAD BALANCING EVALUATION

3.1 Machine Specification

All research evaluations were conducted on a Cray XE6m supercomputer (RAIN) at the Texas A&M University Parasol Lab. Table 3.1 shows the configuration details for this machine.

3.2 Earth Model

In order to study load imbalance, it was necessary to select a relatively complex earth model. Therefore, we built our 3D model based on a synthetic Gulf of Mexico salt dome model (Jain, 2011; Lu et al., 2009; Willis et al., 2006) (Figure 3.1). This model consists of three isotropic regions, two vertical transverse isotropic (VTI) regions, and a salt dome region with a nearby salt canopy region (Table 3.2). The size of this 3D model is (9.00,9.00,6.00) km.

3.3 Input Parameterization

Three main test cases using the salt dome model (Figure 3.1) were evaluated in this research. Each test case had the same input parameters, as shown in Table 3.3.

Configuration	**Value**
Number of computing nodes	24
Total number of cores	576
Processor type	AMD 16 core 64-bit Interlagos 2.1 GHz CPU
Total memory	768GB
OS platform/version	Cray Linux Environment 4.2UP01
File system	36TB LUSTRE parallel file system
Queuing system	Torque/Moab

Table 3.1: Specifications for the RAIN machine, a Cray XE6m (XK7m-200) supercomputer in the Texas A&M University Parasol Lab.

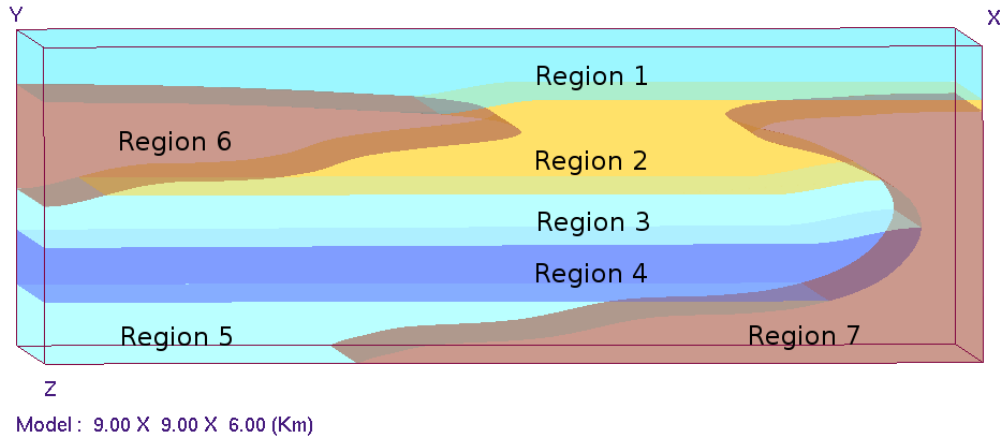


Figure 3.1: A synthetic Gulf of Mexico salt dome model. Regions 1, 4, and 5 are isotropic regions. Regions 2 and 3 are transverse isotropic (VTI) regions. Regions 6 and 7 are salt regions (Jain, 2011).

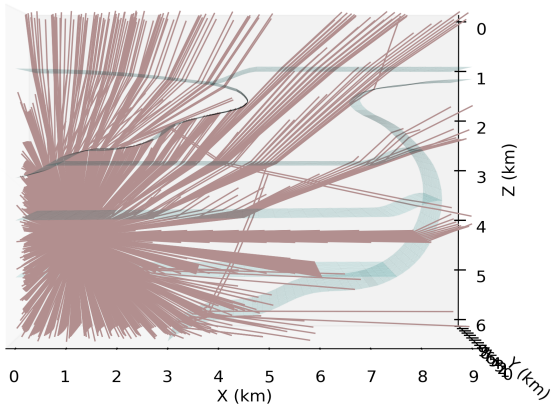
Region number	**Physical properties**
1	$V_p = 3$ km/s, $V_s = 1.73$ km/s, $\rho = 2.5$ g/cm^3
2	$c_{ijkl} = \begin{pmatrix} 20.28 & 13.104 & 15.028 & 0.0 & 0.0 & 0.0 \\ 13.104 & 20.28 & 15.028 & 0.0 & 0.0 & 0.0 \\ 15.028 & 15.028 & 22.542 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 4.498 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 4.498 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 3.588 \end{pmatrix}$ $\rho = 2.4$ g/cm^3
3	$c_{ijkl} = \begin{pmatrix} 25.9 & 6.825 & 7.075 & 0.0 & 0.0 & 0.0 \\ 6.825 & 25.9 & 7.075 & 0.0 & 0.0 & 0.0 \\ 7.075 & 7.075 & 23.775 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 7.325 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 7.325 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 9.525 \end{pmatrix}$ $\rho =2.5$ g/cm^3
4	$V_p = 3.4$ km/s, $V_s = 1.83$ km/s, $\rho = 2.67$ g/cm^3
5	$V_p = 3.8$ km/s, $V_s = 1.9$ km/s, $\rho =2.7$ g/cm^3
6	$V_p = 4.78$ km/s, $V_s = 2.7$ km/s, $\rho = 2.2$ g/cm^3
7	$V_p = 4.78$ km/s, $V_s = 2.7$ km/s, $\rho = 2.2$ g/cm^3

Table 3.2: Regions' physical properties in the salt dome model. V_p is the p-wave velocity, V_s is the s-wave velocity, ρ is the density, and c_{ijkl} is the stiffness tensor.

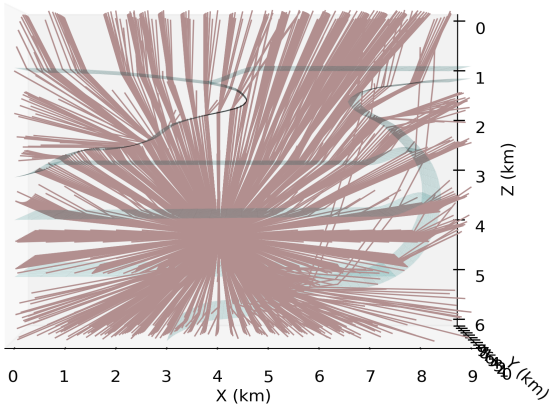
Parameter	Value
Source location	Test cast 1=(1.0,4.5,4.2) km Test cast 2=(4.0,4.5,4.2) km Test cast 3=(7.0,4.5,4.2) km
Ray initialization method	Cubed sphere
Number of initial rays	17x17 in each face of the focal cube Total of 1724 rays
Number of initial ray tubes	16x16 in each face of the focal cube Total of 1536 ray tubes
Wave type	P-wave
Ray captured behavior	Only transmitted rays
Ray tracing step size	0.01 s
Wavefront step size	0.04 s
Interpolation threshold	0.001 s
Coarsening threshold	0.0002 s

Table 3.3: WFC input parameters for all test cases.

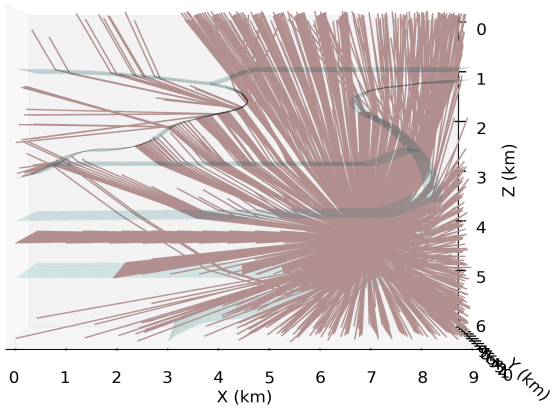
The only difference between the three test cases was the source location. However, all three sources were located in isotropic Region 4. Each test case was executed using the original pWFC with uniform partitioning and combinations of 1, 2, 4, 8, 16, 32, and 64 CPUs. Figure 3.2 shows the preliminary ray tracing (without interpolation) used for the cost (load) estimation. Figures 3.3, 3.4, and 3.5 show the final simulation wavefront snapshots for the three test cases. Supplementary media for this thesis show the WFC method wave simulations and preliminary ray tracing for each test case (Appendix C).



(a)

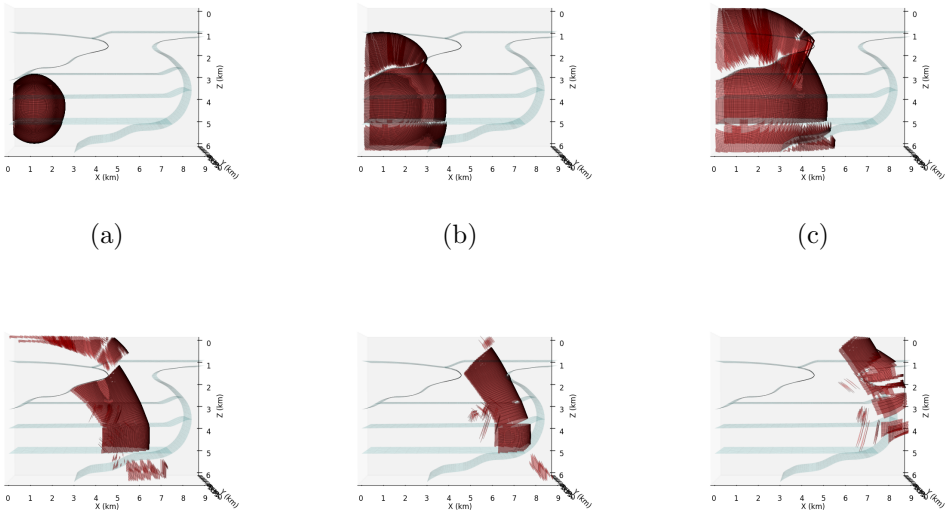


(b)



(c)

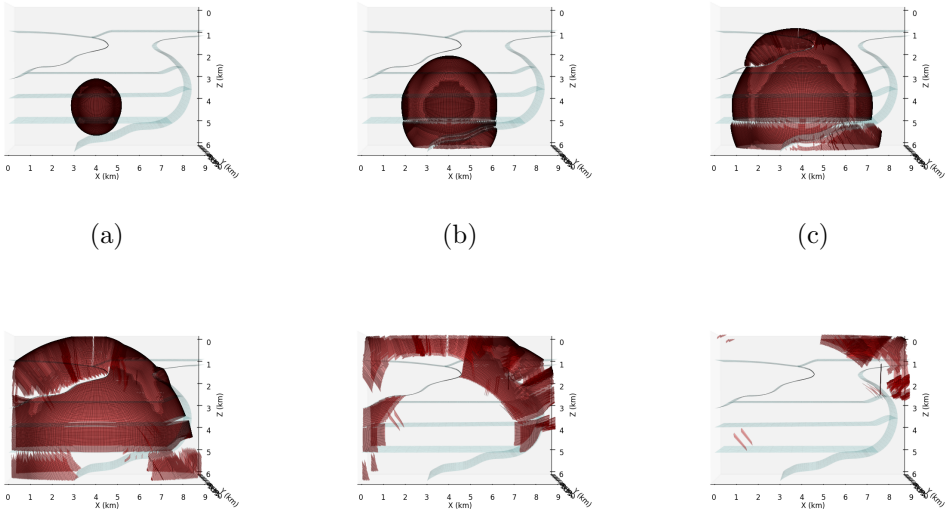
Figure 3.2: Preliminary 3D ray tracing (without interpolation) in salt dome model that was used for cost estimation.(a) Test case 1 with a source located at (1.0,4.5,4.2) km. (b) Test case 2 with a source located at (4.0,4.5,4.2) km. (c) Test case 3 with a source located at (7.0,4.5,4.2) km.



(a) (b) (c)

(d) (e) (f)

Figure 3.3: WFC method simulated wavefront propagation in salt dome model with a source located at (1.0,4.5,4.2) km at time (a) 0.4 s, (b) 0.8, (c) 1.2 s, (d) 1.6 s, (e) 2.0 s, and (f) 2.4 s.



(a) (b) (c)

(d) (e) (f)

Figure 3.4: WFC method simulated wavefront propagation in salt dome model with a source located at (4.0,4.5,4.2) km at time (a) 0.32 seconds, (b) 0.64 seconds, (c) 0.96 seconds, (d) 1.28 seconds, (e) 1.6 seconds, and (f) 1.92 seconds.

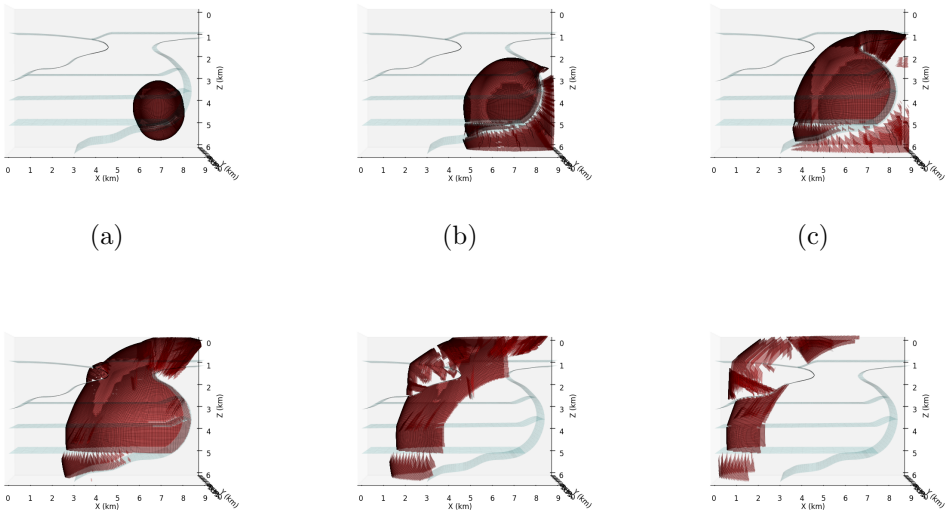


Figure 3.5: WFC method simulated wavefront propagation in salt dome model with a source located at (7.0,4.5,4.2) km at time (a) 0.32 s, (b) 0.64 s, (c) 0.96 s, (d) 1.28 s, (e) 1.6 s, and (f) 1.92 s.

3.4 Weight Prediction Assessment

Using the preliminary ray tracing, we calculated the cost of each cell in the initial wavefront mesh for the three test cases based on the three suggested approaches (Max_E, Max_T and Max_ET). Then we captured the actual load (number of ray tubes) for each CPU, using the original pWFC with uniform partitioning. Following that, and using the original mesh decomposition, we conducted a comparison of the actual and predicted loads for each test case with 2, 4, 8, and 16 CPUs. The purpose of this step was to prove that we had a load imbalance in all three test cases and to evaluate our approaches to weight prediction.

Two examples showing the accuracy of our three approaches to prediction, i.e, the differences between the predicted loads and the actual loads, are shown in Figure 3.6. These plots clearly show different load percentage for the CPUs (load imbalances). They also show that all three approaches to prediction provide a fairly good load estimation. All prediction results are presented in Appendix B.1.

To determine the best approach to weight prediction, we calculated the root-mean-square deviation $RMSD$ between the WFC (actual) and predicted percentages of the loads on each approach for all test cases with combinations of 2, 4, 8, and 16 CPUs, as follows:

$$RMSD = \sqrt{\frac{\sum_{c=0}^{c=n_{cpus}} (P_c - A_c)^2}{n_{cpus}}} \tag{3.1}$$

where n_{cpus} is the number of CPUs (2, 4, 8, or 16), A_c is the WFC (actual) percentage of the load for CPU c and P_c is the predicted percentage of the load for CPU c. Then we calculated the average $RMSD$ for the three approaches to weight prediction for all test cases (Figure 3.7). The original (uniform) load distribution, which assumes that each CPU will have the same load, has the highest $RMSD$ compared to our three

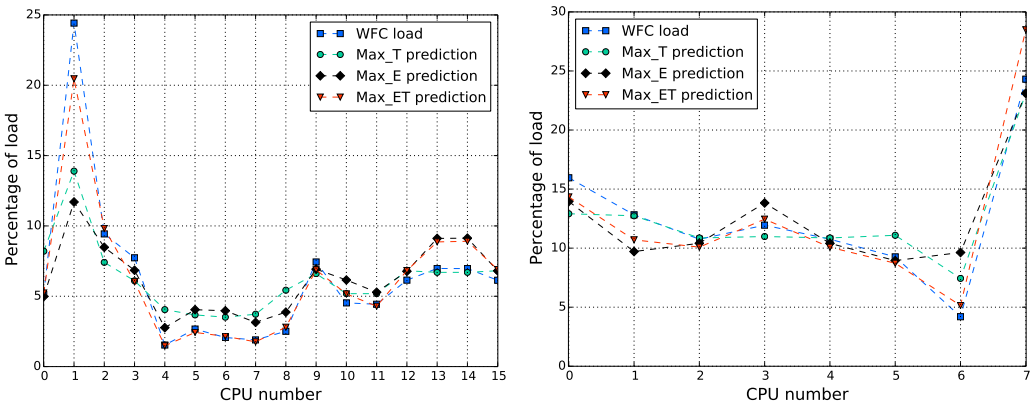
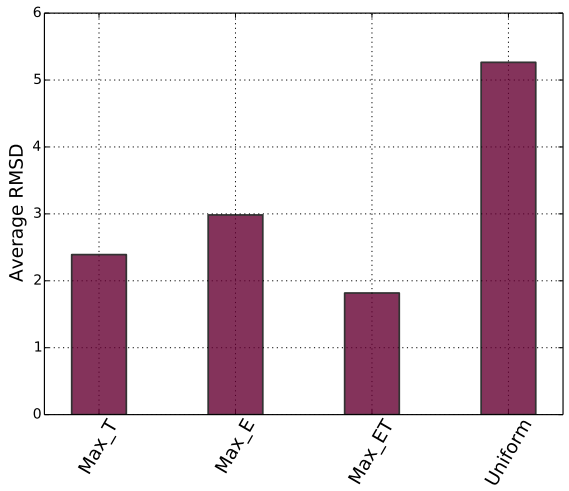


Figure 3.6: Two examples comparing the CPU load percentages for pWFC and the predicted loads based on the preliminary ray tracing used in density prediction for (a) test case 1 with a source located at (1.0,4.5,4.2) km using 16 CPUs, and (b) test case 2 with a source located at (4.0,4.5,4.2) km using 8 CPUs.

approaches. The introduction of ray tube travel time in Max_T and Max_ET results in a better load prediction than Max_E. Overall, these results show that Max_ET has the smallest $RMSD$ from the actual load.

3.5 Non-Uniform Partitioning Evaluation

After choosing Max_ET as the best approach to weight prediction, we executed the modified pWFC with non-uniform partitioning, using combinations of 1, 2, 4, 8, 16, 32, and 64 CPUs for each test case. During the wavefront propagation, in both uniform and non-uniform decomposition, we recorded CPUs' load at each simulation time step. After that, we calculated the load distribution improvement, for each simulation run, by tacking the percentage of dereference between, $RMSD$ of total load percentage using original pWFC implementation and optimal load percentage, and $RMSD$ of total load percentage using our pWFC implementation and optimal load percentage as follows:



(a)

Figure 3.7: The average root-mean-square deviation $RMSD$ between the WFC (actual) and predicted percentage of the loads for each weight prediction approach, using a combination of 2,4,8, and 16 CPUs. Max_T is the maximum travel time weight prediction approach, Max_E is the maximum error weight prediction approach, Max_ET is the maximum travel time and error weight prediction approach, and Uniform is the original (uniform) load distribution.

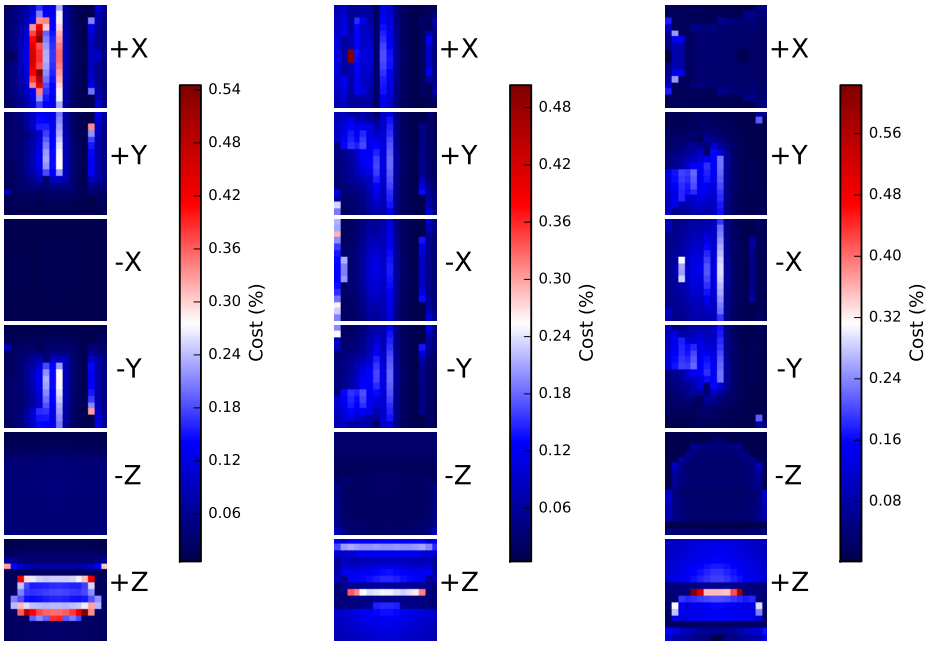
$$improvement = \frac{RMSD_u - RMSD_n}{RMSD_n} * 100 \tag{3.2}$$

where $RMSD_n$ is the root-mean square deviation between uniform decomposition's total CPUs load percentages and optimal load percentages, and $RMSD_u$ is the root-mean square deviation between non-uniform decomposition's total CPUs load percentages and optimal load percentages. In the following, we discuss four examples comparing uniform and non-uniform decomposition by cost prediction. Additional examples are provided in Appendix B.3.

Before discussing the efficiency of the non-uniform decomposition, we show how the cost percentage is distributed among the wavefront mesh cells. Figure 3.8 shows the cost estimation for each cell in the initial wavefront mesh. Using the cubed

sphere initialization method, the initial wavefront consists of six faces (+X, +Y, -X, -Y, +Z, and -Z) that represent a focal cube for ray direction initialization. For example, Figure 3.8 shows a realistic high cost percentage in the +Z face for all three test cases. This high cost arises because of ray tube divergence (a change in the wavefront curvature) in that direction, which increases the difference between the ray tube travel time and the paraxial travel time prediction. Also, rays in this direction travel longer than rays in the -Z direction, as can clearly be seen in Figure 3.2. Another reasonable observation is the symmetry between +Y face and -Y face for all test cases. This symmetry exists because most traced rays in the +Y direction and -Y direction were in the same region, and the sources were located in the middle of Y axis. Therefore, in our new pWFC implementation, the initial wavefront mesh is decomposed into non-equal partitions so that all parts will have roughly equal shares of the cost. This might lead to a load imbalance in the first wavefront propagation steps. However, in later wavefront steps, which usually have greater numbers of rays to process, the ray tubes load should be distributed between the CPUs better than on the uniform decomposition, and this should decrease the turnaround time for the pWFC execution.

The first example comparing uniform and non-uniform decomposition involves test case 2 with a source located at (4.0,4.5,4.2) km. First, we ran this test case on 4 CPUs using the original pWFC (with uniform partitioning). Figure 3.9a shows the initial uniform decomposition distributing the rays/ray tubes among the CPUs. Figure 3.10a shows the amount of the ray tube load that each CPU has during the wavefront propagation. It is obvious that CPU 0 (red color) has a larger load than the other CPUs, and this is clearly reflected in the high predicted cost for that part of the initial mesh (Figures 3.8b and 3.10a). Next we ran our modified pWFC (with non-uniform partitioning) using the same number of CPUs. The non-uniform



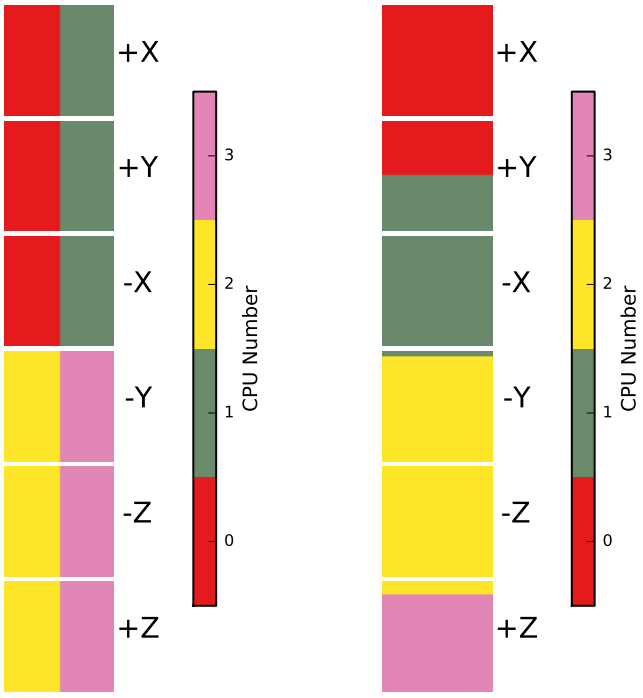
(a) (b) (c)

Figure 3.8: Initial wavefront mesh cost distribution between cells using the Max_ET approach to prediction. The mesh face layout is represented here as it is implemented in the software. Each focal cube face represents the direction of the ray tube. Red indicates a high potential future load and blue indicates a low potential future load. The total cost for each wavefront mesh is 100%. (a) Test case 1 with a source located at (1.0,4.5,4.2) km, (b) test case 2 with a source located at (4.0,4.5,4.2) km, and (c) test case 3 with a source located at (7.0,4.5,4.2) km.

decomposition of the initial wavefront mesh is based on the ray density predictions (Figure 3.8b) using preliminary ray tracing (Figure 3.2b). Thus, Figure 3.9b shows different partition sizes for the CPUs, and Figure 3.10b shows the amount of the ray tube load that each CPU has during the wavefront propagation. This execution achieves a reduction of more than 1000 ray tubes for the average CPUs load difference (Figure 3.11), and improves the total CPUs load distribution by more than 45% (Figure 3.12). pWFC wavefront simulation results, which are exactly the same for both versions, are shown in Figure 3.4.

Also, the second interesting example comes from test case 2 with a source located at (4.0,4.5,4.2) km. First, we ran this test case with 16 CPUs using the original pWFC (uniform partitioning). Figure 3.13a shows the initial uniform decomposition distributing rays/ray tubes to the CPUs. Figure 3.14a shows the amount of the ray tube load that each CPU has during the wavefront propagation. It is obvious that CPU 1 (purple color) has a much larger load than the other CPUs, and this is clearly seen in the high predicted cost in that part of the initial mesh (Figures 3.8b and 3.14b). In contrast, CPU 11 (yellow color) has a very small load, which is also reflected in our cost estimation. Next, we ran our modified pWFC (non-uniform partitioning) with the same number of CPUs. The non-uniform decomposition of the initial wavefront mesh is based on ray density prediction (Figure 3.8b) using preliminary ray tracing (Figure 3.2b). Thus, Figure 3.13b shows that some CPUs have very small partition sizes while others have larger sizes. Figure 3.14b shows the amount of the ray tube load that each CPU has during the wavefront propagation. This test achieves more than 57% improvement in overall load distribution (Figure 3.16), and a large reduction of more than 2500 ray tubes in the average CPUs load (Figure 3.15).

Third example comparing uniform and non-uniform decomposition involves test case 1 with a source located at (1.0,4.5,4.2) km. First, we ran this test case on



(a) (b)

Figure 3.9: Initial wavefront mesh partitioning for test case 2 with a source located at (4.0,4.5,4.2) km using 4 CPUs. The mesh face layout is represented here as it is implemented in the software. Each focal cube face represents the direction of the ray tube. (a) Original pWFC implementation using uniform partitioning. (b) Our modified pWFC implementation using non-uniform partitioning.

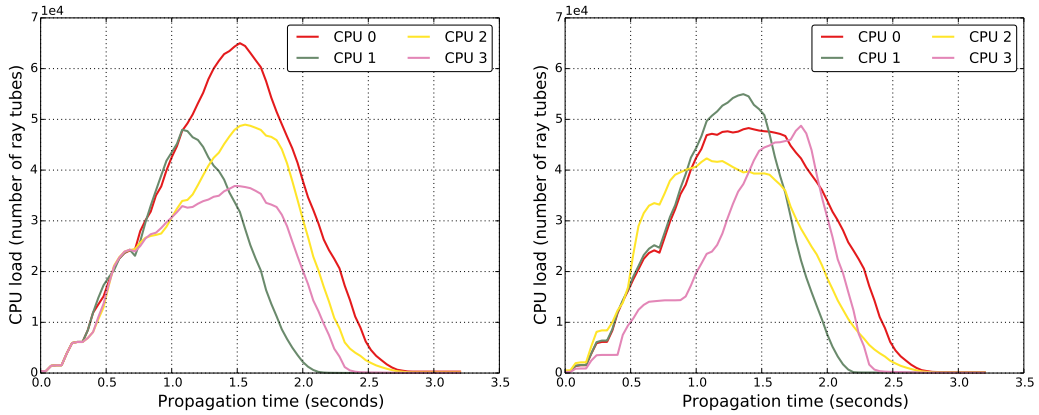


Figure 3.10: CPU load profiles during wavefront propagation for test case 2 with a source located at (4.0,4.5,4.2) km using 4 CPUs. The colors and CPU numbers can be cross-referenced with the wavefront mesh partitioning shown in Figure 3.9. (a) Original pWFC implementation using uniform partitioning. (b) Our modified pWFC implementation using non-uniform partitioning.

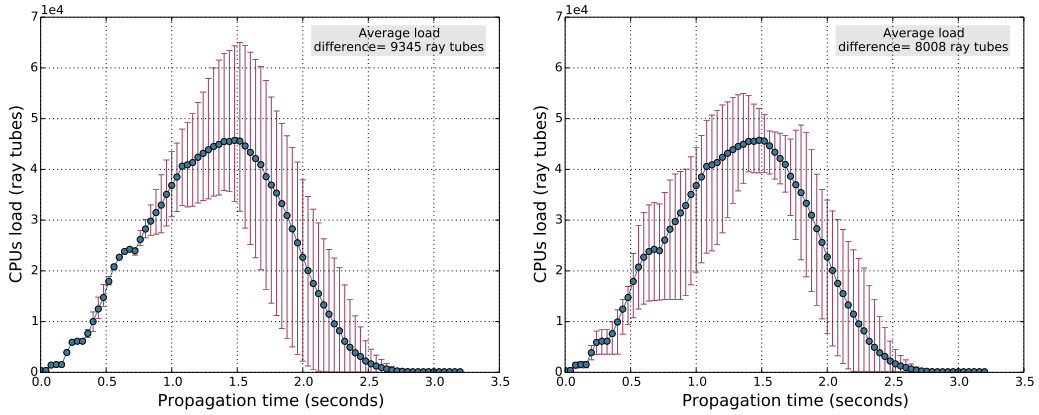
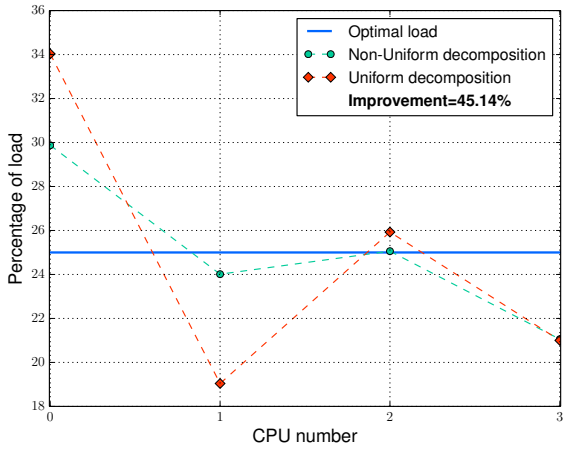


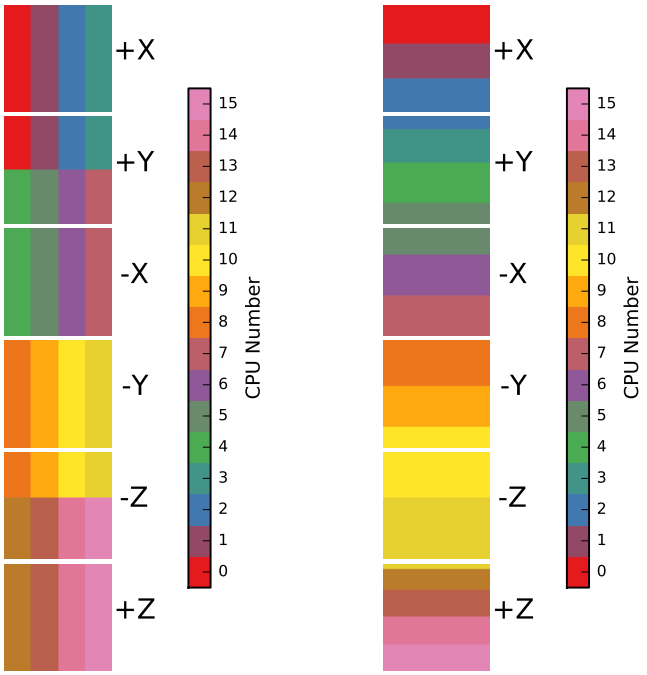
Figure 3.11: Difference between CPU loads (ray tubes) during wavefront propagation for test case 2 with a source located at (4.0,4.5,4.2) km using 4 CPUs. Green circles indicates average load, and red bars show maximum and minimum load. (a) Original pWFC implementation using uniform partitioning. (b) Our modified pWFC implementation using non-uniform partitioning.



(a)

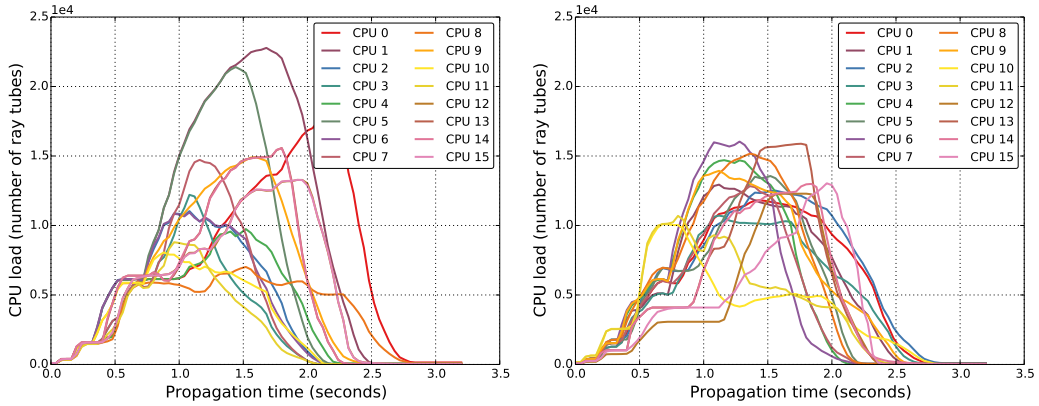
Figure 3.12: Total CPUs' percentage of load distribution of uniform and non-uniform decomposition for test case 2 with a source located at (4.0,4.5,4.2) km using 4 CPUs. The percentage of improvement is calculated using equation 3.2.

4 CPUs using the original pWFC (with uniform partitioning). Figure 3.17a shows the initial uniform decomposition distributing the rays/ray tubes among the CPUs. Figure 3.18a shows the amount of the ray tube load that each CPU has during the wavefront propagation. Next we ran our modified pWFC (with non-uniform partitioning) using the same number of CPUs. The non-uniform decomposition of the initial wavefront mesh is based on the ray density predictions (Figure 3.8a) using preliminary ray tracing (Figure 3.2a). Thus, Figure 3.17b shows different partition sizes for the CPUs, and Figure 3.18b shows the amount of the ray tube load that each CPU has during the wavefront propagation. In contrast to the first two examples, here the non-uniform partitioning results in higher load differences compared to uniform partitioning (Figure 3.19). For instance, CPU 2 (yellow color) has a high load compared to other CPUs (Figure 3.18b), which exists only in the first simulation steps. This behavior appears because CPU 2 is responsible for tracing a large portion of the wavefront mesh with rays that terminated sooner than others (-X and -Z



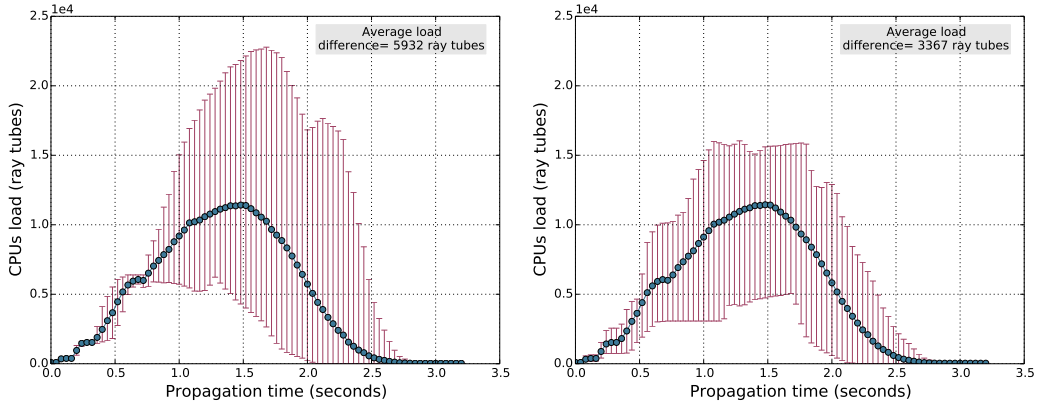
(a) (b)

Figure 3.13: Initial wavefront mesh partitioning for test case 2 with a source located at (4.0,4.5,4.2) km using 16 CPUs. The mesh face layout is represented here as it is implemented in the software. Each focal cube face represents the direction of the ray tube. (a) Original pWFC implementation using uniform partitioning. (b) Our modified pWFC implementation using non-uniform partitioning.



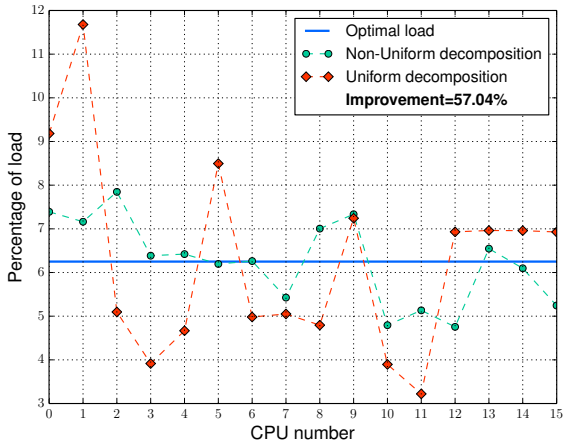
(a) (b)

Figure 3.14: CPU load profiles during wavefront propagation for test case 2 with a source located at (4.0,4.5,4.2) km using 16 CPUs. The colors and CPU numbers can be cross-referenced with the wavefront mesh partitioning shown in Figure 3.9. (a) Original pWFC implementation using uniform partitioning. (b) Our modified pWFC implementation using non-uniform partitioning.



(a) (b)

Figure 3.15: Difference between CPU loads (ray tubes) during wavefront propagation for test case 2 with a source located at (4.0,4.5,4.2) km using 16 CPUs. Green circles indicates average load, and red bars show maximum and minimum load. (a) Original pWFC implementation using uniform partitioning. (b) Our modified pWFC implementation using non-uniform partitioning.

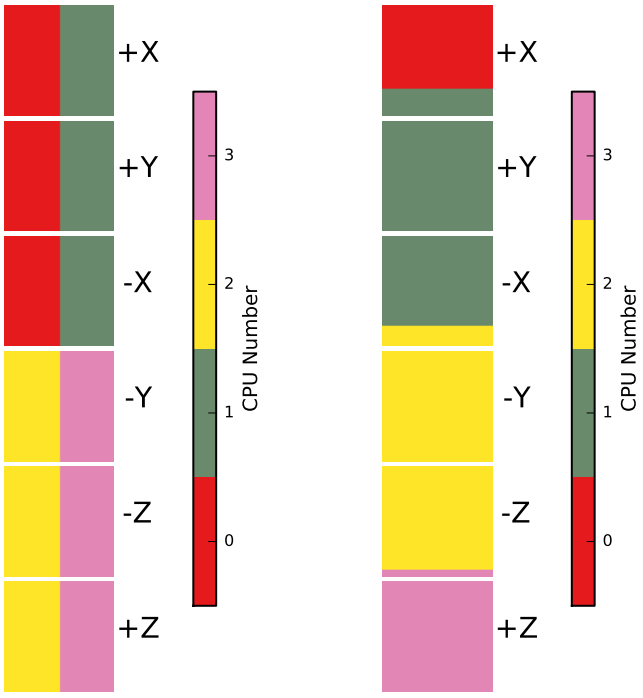


(a)

Figure 3.16: Total CPUs' percentage of load distribution of uniform and non-uniform decomposition for test case 2 with a source located at (4.0,4.5,4.2) km using 16 CPUs. The percentage of improvement is calculated using equation 3.2.

faces) (Figures 3.17b and 3.2a). Therefore, even though the total load distribution is improved by more than 55% (Figure 3.20), in this case the average difference in number of ray tubes between CPUs increases (Figure 3.19). This should also result in a slight decrease in performance compared to the original pWFC implementation. pWFC wavefront simulation results, which are exactly the same for both versions, are shown in Figure 3.3.

The fourth interesting example comes from test case 1 too. First, we ran this test case with 16 CPUs using the original pWFC (uniform partitioning). Figure 3.21a shows the initial uniform decomposition distributing rays/ray tubes to the CPUs. Figure 3.22a shows the amount of the ray tube load that each CPU has during the wavefront propagation. It is obvious that CPU 1 (purple color) has a much larger load than the other CPUs, and this is clearly seen in the high predicted cost in that part of the initial mesh (Figures 3.8a and 3.22b). In contrast, CPU 4 (green color) has a very small load, which is also reflected in our cost estimation. Next,



(a) (b)

Figure 3.17: Initial wavefront mesh partitioning for test case 1 with a source located at (1.0,4.5,4.2) km using 4 CPUs. The mesh face layout is represented here as it is implemented in the software. Each focal cube face represents the direction of the ray tube. (a) Original pWFC implementation using uniform partitioning. (b) Our modified pWFC implementation using non-uniform partitioning.

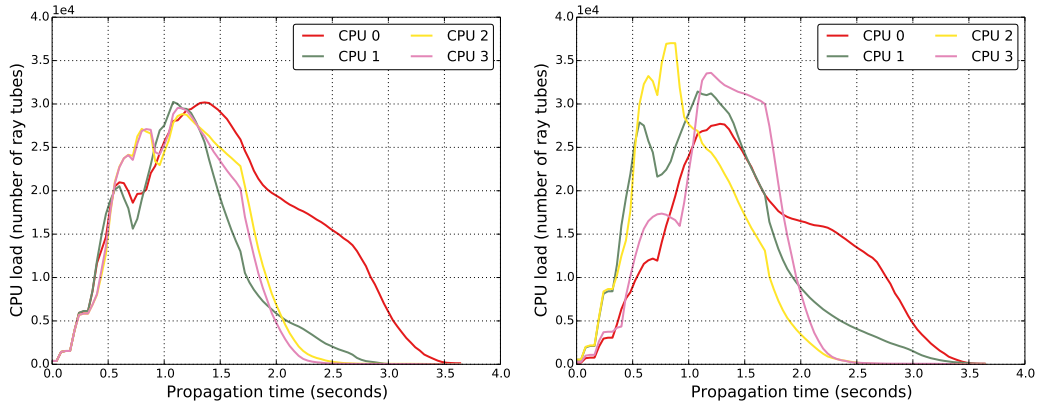


Figure 3.18: CPU load profiles during wavefront propagation for test case 1 with a source located at (1.0,4.5,4.2) km using 4 CPUs. The colors and CPU numbers can be cross-referenced with the wavefront mesh partitioning shown in Figure 3.9. (a) Original pWFC implementation using uniform partitioning. (b) Our modified pWFC implementation using non-uniform partitioning.

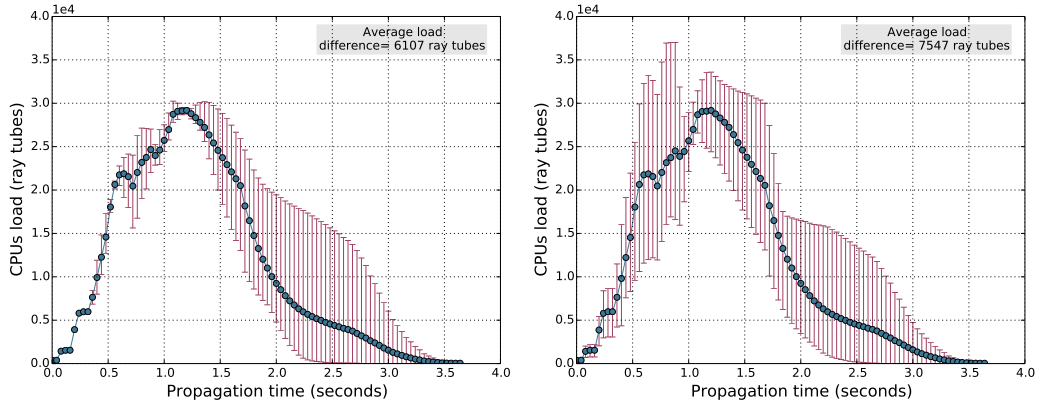
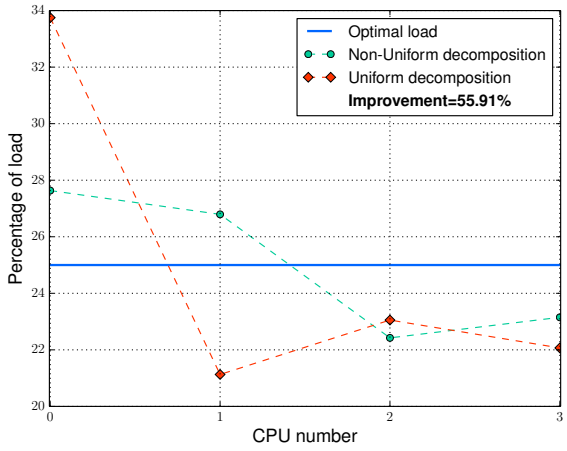


Figure 3.19: Difference between CPU loads (ray tubes) during wavefront propagation for test case 1 with a source located at (1.0,4.5,4.2) km using 4 CPUs. Green circles indicates average load, and red bars show maximum and minimum load. (a) Original pWFC implementation using uniform partitioning. (b) Our modified pWFC implementation using non-uniform partitioning.



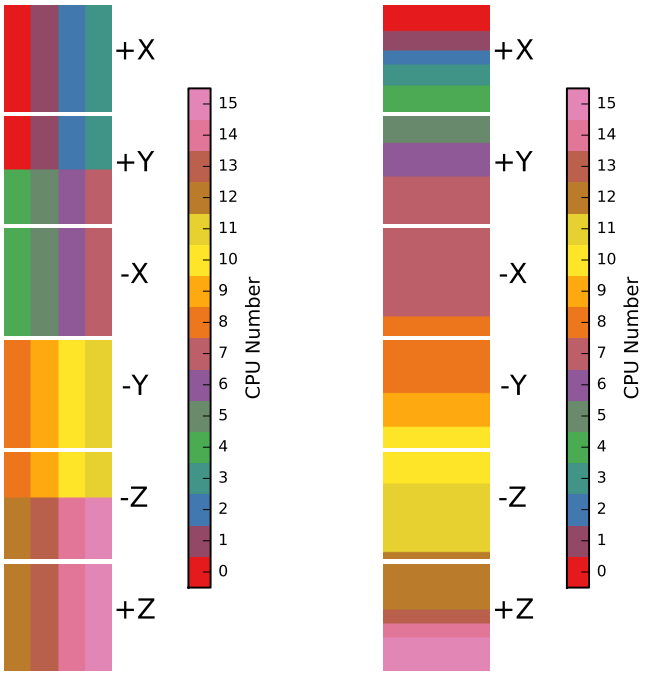
(a)

Figure 3.20: Total CPUs' percentage of load distribution of uniform and non-uniform decomposition for test case 1 with a source located at (1.0,4.5,4.2) km using 4 CPUs. The percentage of improvement is calculated using equation 3.2.

we ran our modified pWFC (non-uniform partitioning) with the same number of CPUs. The non-uniform decomposition of the initial wavefront mesh is based on ray density prediction (Figure 3.8a) using preliminary ray tracing (Figure 3.2a). Thus, Figure 3.21b shows that some CPUs have very small partition sizes while others have larger sizes. Figure 3.22b shows the amount of the ray tube load that each CPU has during the wavefront propagation. This test achieves more than 74% improvement in overall load distribution (Figure 3.24), and great reduction of more than 5000 ray tubes in the average CPUs load (Figure 3.23). Third and fourth examples show a significant variation of load balancing, although they are based on the same test case (source location).

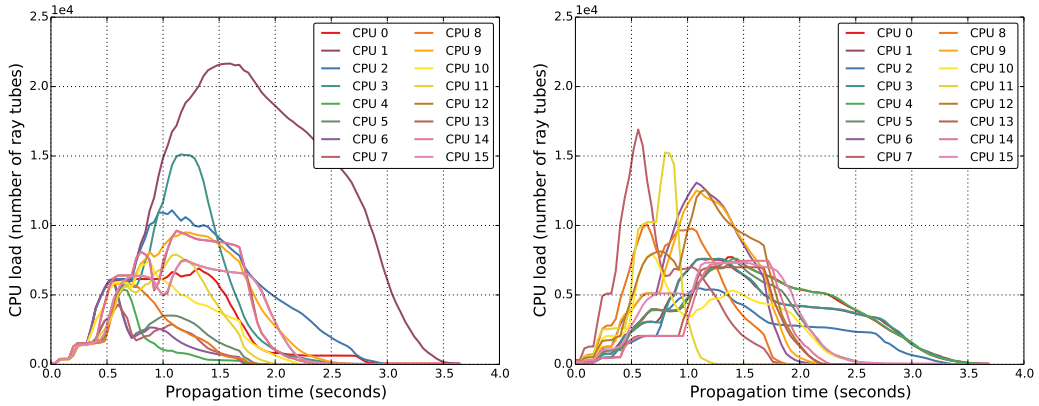
3.6 Final Performance Results

As discussed, both versions of pWFC were executed with combinations of 1, 2, 4, 8, 16, 32, and 64 CPUs. In order to accurately measure the performance, each



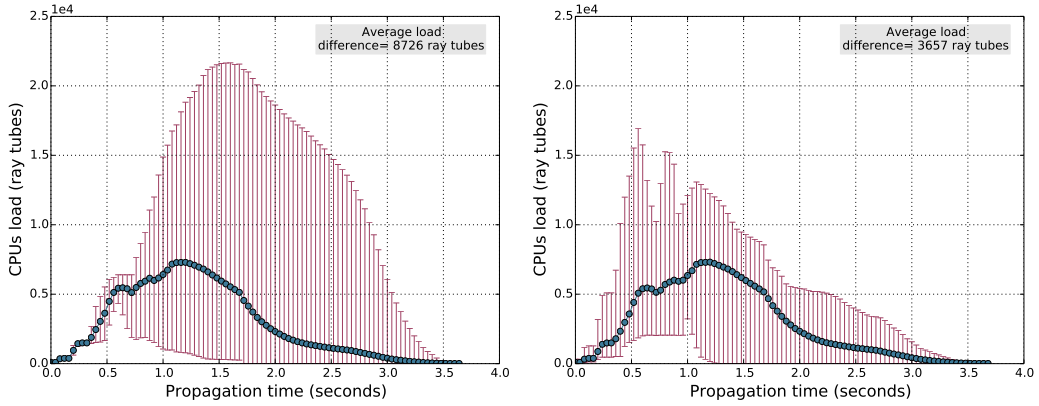
(a) (b)

Figure 3.21: Initial wavefront mesh partitioning for test case 1 with a source located at (1.0,4.5,4.2) km using 16 CPUs. The mesh face layout is represented here as it is implemented in the software. Each focal cube face represents the direction of the ray tube. (a) Original pWFC implementation using uniform partitioning. (b) Our modified pWFC implementation using non-uniform partitioning.



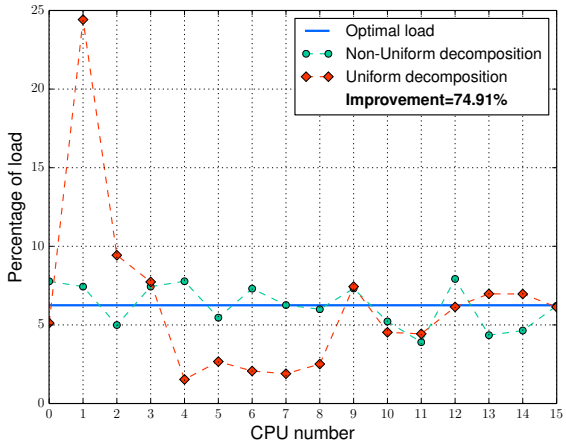
(a) (b)

Figure 3.22: CPU load profiles during wavefront propagation for test case 1 with a source located at (1.0,4.5,4.2) km using 16 CPUs. The colors and CPU numbers can be cross-referenced with the wavefront mesh partitioning shown in Figure 3.9. (a) Original pWFC implementation using uniform partitioning. (b) Our modified pWFC implementation using non-uniform partitioning.



(a) (b)

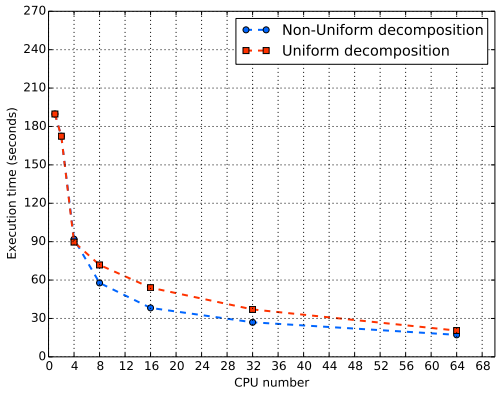
Figure 3.23: Difference between CPU loads (ray tubes) during wavefront propagation for test case 1 with a source located at (1.0,4.5,4.2) km using 16 CPUs. Green circles indicates average load, and red bars show maximum and minimum load. (a) Original pWFC implementation using uniform partitioning. (b) Our modified pWFC implementation using non-uniform partitioning.



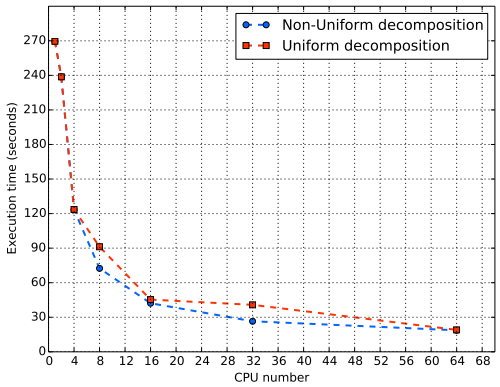
(a)

Figure 3.24: Total CPUs' percentage of load distribution of uniform and non-uniform decomposition for test case 1 with a source located at (1.0,4.5,4.2) km using 16 CPUs. The percentage of improvement is calculated using equation 3.2.

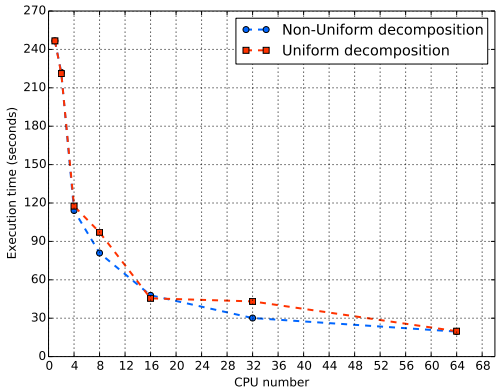
simulation run were executed five times. However, the execution time difference between the five runs is negligible compared to the absolute total execution time. For most of the cases, the non-uniform initial decomposition shows a smaller turnaround time (Figure 3.25). Therefore, we can see in Figure 3.26 that our modified pWFC achieves better scalability. One interesting observation here is that original pWFC implementation has unstable scalability compared to our implementation. This unsuitability appears because uniform decomposition might, by a chance, lead to a good load balancing. However, in our implementation, the initial wavefront mesh in partitioned based on a systematic criterion (future ray density), results in more stable scalability.



(a)

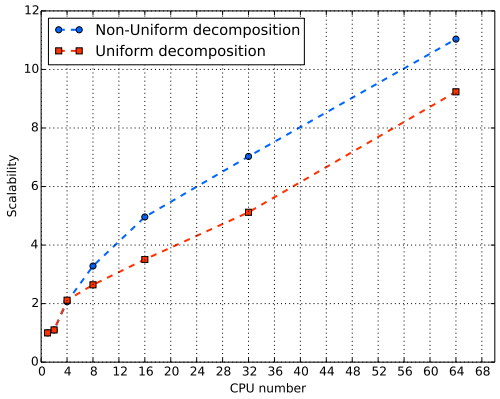


(b)

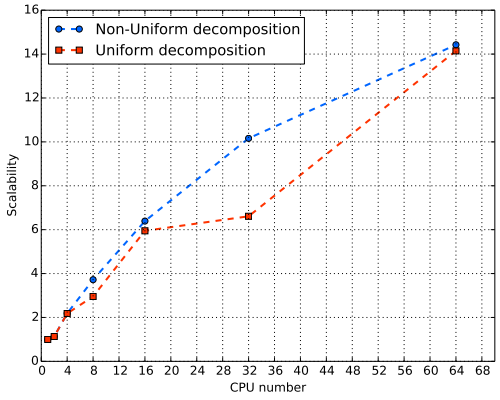


(c)

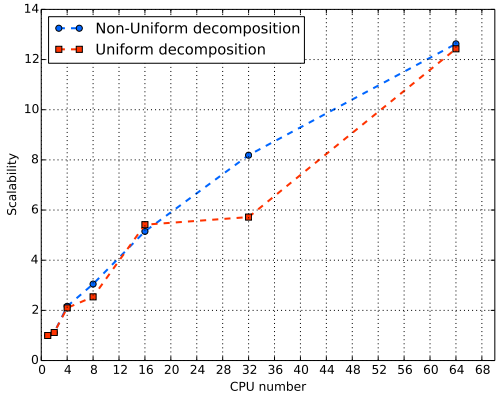
Figure 3.25: Comparison of the total execution times for the original pWFC (uniform partitioning) and our modified pWFC (uniform partitioning) using 1, 2, 4, 8, 16, 32, and 64 CPUs for (a) test case 1 with a source located at (1.0,4.5,4.2) km, (b) test case 2 with a source located at (4.0,4.5,4.2) km, and (c) test case 3 with a source located at (7.0,4.5,4.2) km.



(a)



(b)



(c)

Figure 3.26: Comparison of the scalability of the original pWFC (uniform partitioning) and our modified pWFC (uniform partitioning) using 1, 2, 4, 8, 16, 32, and 64 CPUs for (a) test case 1 with a source located at (1.0,4.5,4.2) km, (b) test case 2 with a source located at (4.0,4.5,4.2) km, and (c) test case 3 with a source located at (7.0,4.5,4.2) km.

4. DISCUSSION

To evaluate our proposed approaches to weight estimation and to analyze the behavior of our proposed non-uniform mesh partitioning, we tested three cases based on a synthetic salt dome model.

We evaluated our proposed approaches to weight prediction in order to prove the existence of a load imbalance and to assess the accuracy of each approach's estimations. All three test cases showed a clear load imbalance for different CPU combinations. Two main variables contribute to the weight predictions in all three approaches: the ray tube travel time and error. The ray tube error, which is based on paraxial travel time prediction, is suggested due to its importance in indicating the probability of future ray interpolation. However, the use of a combination of the two variables, travel time and error, rather than the one of them proved to be more reliable for the load estimation. This establishes the importance of using travel time in the predictions, because some ray tubes may be terminated sooner than others and this leads to load imbalances.

We compared uniform and non-uniform mesh decomposition to show the efficiency of our cost (load) prediction method. We proved its effectiveness by using a simple 2D block (rectangular) decomposition technique that decomposes the initial wavefront mesh into parts that have roughly equal costs regardless of their sizes. In the future, it would be interesting to investigate the behavior of non-rectangular decomposition. However, CPU communication costs should also be considered in this situation.

Generally, seismic modeling applications use multiple sources that are distributed over the model. However, it is not efficient to apply the preliminary ray tracing for

every source, and it would therefore be preferable to sample a subset of the sources. A computational cost for the initial meshes can then be estimated, and this prediction can be interpolated and applied though subvolumes. This process will be very helpful in predicting the complexity of the earth model and efficiently estimating the future load.

In few simulation runs we noticed a slight decrease in performance of our pWFC implementation compared to the original implementation. This decrease appeared because uniform decomposition can sometimes lead fortuitously to a good ray tube distribution. However, the final performance results showed a stable scalability improvement over the original implementation when applying the load balancing approach.

All the simulation runs showed that our implementation resulted in high improvement of CPUs load distribution, which proves that our ray density prediction can estimate the overall load distribution. However, results also showed this improvement dose not guarantee a complete load balancing between CPUs, because they might still have a considerable amount of load variations in some of the wavefront simulation steps as the wavefront propagates through the model. Because we are using both the maximum ray tube travel time and error in our cost estimation, these results are not surprising. However, since we know the ray tube travel time and error at each wavefront time steps, we could apply the criterion during propagation to readjust the load balance (distribution of ray tubes) between processors as the wavefront propagates. This could therefore serve as the basis of a dynamic load balancing approach.

The present study showed how, even with static load balancing, better performance can be achieved. This result can be used to accomplish our ultimate goal, which is a fully load-balanced pWFC. Dynamic load balancing entails a huge amount

of CPU communication and work redistribution, and our proposed cost estimation can provide an indication of how much future load imbalance might be expected, which can be used to determine whether dynamic load balancing will be worthwhile. Additionally, a mixture of the two methods of load balancing, dynamic and static, might result in better performance.

5. CONCLUSION

This study has implemented a new static load balancing algorithm for pWFC. The method consists of two major parts: density load estimation and non-uniform decomposition. We first evaluated different approaches for predicting future loads. Next, we investigated a non-uniform partitioning of the initial wavefront mesh and tested our new algorithm on three test cases based on a synthetic salt dome model. Our results show that our new algorithm achieves better and stable scalability in most of the cases. Overall, this research has studied the behavior of pWFC load imbalances and should help in deciding upon the best strategy for fully load-balanced pWFC applications.

REFERENCES

Alaei, B., 2012. Seismic modeling of complex geological structures. In: Kanao, M. (Ed.), Seismic Waves - Research and Analysis. InTech, Rijeka, Croatia, Ch. 11, pp. 213–233.

URL http://dx.doi.org/10.5772/29423

An, P., Jula, A., Rus, S., Saunders, S., Smith, T., Tanase, G., Thomas, N., Amato, N., Rauchwerger, L., 2003. STAPL: An adaptive, generic parallel C++ library. In: Dietz, H. (Ed.), Languages and Compilers for Parallel Computing. Vol. 2624 of Lecture Notes in Computer Science. Springer, New York, NY, pp. 193–208.

URL http://dx.doi.org/10.1007/3-540-35767-X_13

Bohlen, T., 2002. Parallel 3-D viscoelastic finite difference seismic modelling. Computers & Geosciences 28 (8), 887 – 899.

URL http://dx.doi.org/10.1016/S0098-3004(02)00006-7

Buss, A., Harshvardhan, Papadopoulos, I., Pearce, O., Smith, T., Tanase, G., Thomas, N., Xu, X., Bianco, M., Amato, N. M., Rauchwerger, L., 2010. STAPL: Standard template adaptive parallel library. In: Proceedings of the 3rd Annual Haifa Experimental Systems Conference. SYSTOR '10. ACM, New York, NY, pp. 14:1–14:10.

URL http://dx.doi.org/10.1145/1815695.1815713

Carcione, J., Herman, G., ten Kroode, A., 2002. Seismic modeling. Geophysics 67 (4), 1304–1325.

URL http://dx.doi.org/10.1190/1.1500393

Chambers, K., Kendall, J.-M., 2008. A practical implementation of wavefront construction for 3-D isotropic media. Geophysical Journal International 173 (3), 1030–

1038.

URL http://dx.doi.org/10.1111/j.1365-246X.2008.03790.x

Chen, B., 2011. Efficient smoothing and interpolation of velocity models for seismic wavefront construction algorithms. M.Sc. Thesis, Texas A&M University, College Station, TX, 46 pp.

URL http://hdl.handle.net/1969.1/ETD-TAMU-2011-08-9761

Coman, R., Gajewski, D., 2001. Estimation of multivalued arrivals in 3D models using wavefront ray tracing. In: SEG Technical Program Expanded Abstracts 2001. Society of Exploration Geophysicists, Tulsa, OK, pp. 1265–1268.

URL http://dx.doi.org/10.1190/1.1816324

Fehler, M. C., Huang, L., 2002. Modern imaging using seismic reflection data. Annual Review of Earth and Planetary Sciences 30 (1), 259–284.

URL http://dx.doi.org/10.1146/annurev.earth.30.091201.140909

Fidel, A., Jacobs, S. A., Sharma, S., Amato, N. M., Rauchwerger, L., 2014. Using load balancing to scalably parallelize sampling-based motion planning algorithms. In: Proceedings of IEEE 28th International Parallel and Distributed Processing Symposium. IEEE Computer Society, Los Alamitos, CA, pp. 573–582.

URL https://parasol.tamu.edu/publications/download.php?file_id=865

Gajewski, D., Coman, R., Vanelle, C., 2002. Amplitude preserving Kirchhoff migration: A traveltime based strategy. Studia Geophysica et Geodaetica 46 (2), 193–211.

URL http://dx.doi.org/10.1023/A%3A1019849919186

Gibson, R., Durussel, V., Lee, K., 2005. Modeling and velocity analysis with a wavefront-construction algorithm for anisotropic media. Geophysics 70 (4), T63–T74.

URL http://dx.doi.org/10.1190/1.1988188

Gibson Jr, R., 2000. Ray tracing by wavefront construction for anisotropic media. In: SEG Technical Program Expanded Abstracts 2000. Society of Exploration Geophysicists, Tulsa, OK, pp. 2305–2308.
URL http://dx.doi.org/10.1190/1.1815919

Gjøystdal, H., Iversen, E., Laurain, R., Lecomte, I., Vinje, V., Åstebøl, K., 2002. Review of ray theory applications in modelling and imaging of seismic data. Studia Geophysica et Geodaetica 46 (2), 113–164.
URL http://dx.doi.org/10.1023/A:1019893701439

Gjøystdal, H., Iversen, E., Lecomte, I., Kaschwich, T., Drottning, Å., Mispel, J., 2007. Improved applicability of ray tracing in seismic acquisition, imaging, and interpretation. Geophysics 72 (5), SM261–SM271.
URL http://dx.doi.org/10.1190/1.2736515

Grunberg, M., Genaud, S., Mongenet, C., 2004. Seismic ray-tracing and earth mesh modeling on various parallel architectures. The Journal of Supercomputing 29 (1), 27–44.
URL http://dx.doi.org/10.1023/B%3ASUPE.0000022571.28175.e9

Hanxleden, R. V., Scott, L., 1991. Load balancing on message passing architectures. Journal of Parallel and Distributed Computing 13 (3), 312 – 324.
URL http://dx.doi.org/10.1016/0743-7315(91)90078-N

Jain, T. K., 2011. Parallel seismic ray tracing. Unpublished M.Sc. Thesis, Texas A&M University, College Station, TX, 75 pp.

Kaschwich, T., 2006. Traveltime computation and migration in anisotropic media. Ph.D. Dissertation, Universität Hamburg, Hamburg, Germany, 142 pp.
URL http://ediss.sub.uni-hamburg.de/volltexte/2007/3192

Komatitsch, D., Gddeke, D., Erlebacher, G., Micha, D., 2010. Modeling the propagation of elastic waves using spectral elements on a cluster of 192 GPUs. Computer

Science - Research and Development 25 (1-2), 75–82.

URL http://dx.doi.org/10.1007/s00450-010-0109-1

Lee, K., Gibson, R., 2007. An improved mesh generation scheme for the wavefront construction method. Geophysics 72 (1), T1–T8.

URL http://dx.doi.org/10.1190/1.2399366

Lee, K. J., 2005. Efficient ray tracing algorithms based on wavefront construction and model based interpolation method. Ph.D. Dissertation, Texas A&M University, College Station, TX, 125 pp.

URL http://hdl.handle.net/1969.1/3771

Lu, R., Willis, M. E., Campman, X. H., Toksz, M. N., 2009. Evaluation of elastodynamic interferometric redatuming: a synthetic study on salt dome flank imaging. Geophysical Journal International 176 (3), 889–896.

URL http://dx.doi.org/10.1111/j.1365-246X.2008.04019.x

Mohammadzaheri, A., Sadeghi, H., Hosseini, S. K., Navazandeh, M., 2013. DISRAY: A distributed ray tracing by map-reduce. Computers & Geosciences 52 (0), 453 – 458.

URL http://dx.doi.org/10.1016/j.cageo.2012.10.009

Musser, D. R., Derge, G. J., Saini, A., 2001. STL Tutorial and Reference Guide: C++ Programming with the Standard Template Library, 2nd Edition. Addison-Wesley Professional, Indianapolis, IN, 560pp.

Szostek, K., Leniak, A., 2012. Parallelization of the seismic ray trace algorithm. In: Wyrzykowski, R., Dongarra, J., Karczewski, K., Waniewski, J. (Eds.), Parallel Processing and Applied Mathematics. Vol. 7204 of Lecture Notes in Computer Science. Springer, New York, NY, pp. 411–418.

URL http://dx.doi.org/10.1007/978-3-642-31500-8_42

Červený, V., 2005. Seismic Ray Theory. Cambridge University Press, Cambridge,

U.K., 724pp.

URL http://bit.ly/1k7qoOY

Vinje, V., Åstebøl, K., Iversen, E., Gjøystdal, H., 1999. 3-D ray modeling by wavefront construction in open models. Geophysics 64 (6), 1912–1919.

URL http://dx.doi.org/10.1190/1.1444697

Vinje, V., Iversen, E., Åstebøl, K., Gjøystdal, H., 1996. Estimation of multivalued arrivals in 3D models using wavefront construction—Part I1. Geophysical Prospecting 44 (5), 819–842.

URL http://dx.doi.org/10.1111/j.1365-2478.1996.tb00175.x

Vinje, V., Iversen, E., Gjøystdal, H., 1993. Traveltime and amplitude estimation using wavefront construction. Geophysics 58 (8), 1157–1166.

URL http://dx.doi.org/10.1190/1.1443499

Willis, M., Lu, R., Campman, X., Nafi Toksz, M., Zhang, Y., Hoop, M., 2006. A novel application of time-reversed acoustics: Salt-dome flank imaging using walkaway vsp surveys. Geophysics 71 (2), A7–A11.

URL http://dx.doi.org/10.1190/1.2187711

APPENDIX A

KEY RESULTS FOR WFC

The WFC method is based on applying a high-frequency approximation to the elastic wave equation. It simulates seismic wave propagation by tracing ray fields rather than individual rays. It begins with an initial sparse set of rays and interpolates new rays whenever an accuracy criterion is violated.

A.1 Seismic Ray Tracing

Ray paths, travel times, and amplitudes for general anisotropic media are calculated by solving a set of ordinary differential equations (ODEs) of the following form (Gibson et al., 2005):

$$\frac{dx_i}{d\tau} = a_{ijkl} p_l g_j g_k \tag{A.1}$$

$$\frac{dp_i}{d\tau} = -\frac{1}{2}\frac{da_{ijkl}}{dx_i} p_n p_l g_j g_k \tag{A.2}$$

$$a_{ijkl} = \frac{c_{ijkl}}{\rho}, \tag{A.3}$$

where τ is the travel time, x_i is a spatial coordinate component, a_{ijkl} is a density normalized elastic modulus (stiffness tensor), p_i is a component of the slowness vector, and g_i is a component of the particle-motion vector. In the present project, these ODEs (A.1 and A.2) are solved for each ray using fifth-order Runge Kutta methods.

A.2 Predicted Paraxial Travel Time

In this project, the accuracy criterion used for interpolating new rays is based on the predicted paraxial travel time. This is a second-order Taylor series expansion of

travel time from a known location on ray x to a nearby location y in the following form (Lee and Gibson, 2007):

$$\tau(\mathbf{x}) \approx \tau(\mathbf{y}) + p_i(x_i - y_i) + \frac{1}{2}\frac{\partial^2\tau}{\partial x_i \partial x_j}(x_i - y_i)(x_j - y_j), \tag{A.4}$$

where the slowness vector components p_i are the first derivatives of the travel time field. The second derivatives have the following form:

$$\frac{\partial^2\tau}{\partial x_i \partial x_j} = \frac{p_i}{\gamma_k}\left(\frac{\partial x_j}{\partial \gamma_k}\right)^{-1}. \tag{A.5}$$

Here the γ_k are ray coordinates, which are the travel time τ and azimuthal and inclination takeoff angles of the rays. This method is based on predicting the travel time for a point located diagonally to the reference ray. If the difference between the predicted and actual travel times exceeds a predefined threshold, new rays are interpolated in the mesh cell. The predicted paraxial travel time is also used to calculate the travel time for new rays and the travel time for receivers.

APPENDIX B

SUPPLEMENTARY FIGURES

B.1 Actual Load vs. Weight Prediction Approaches

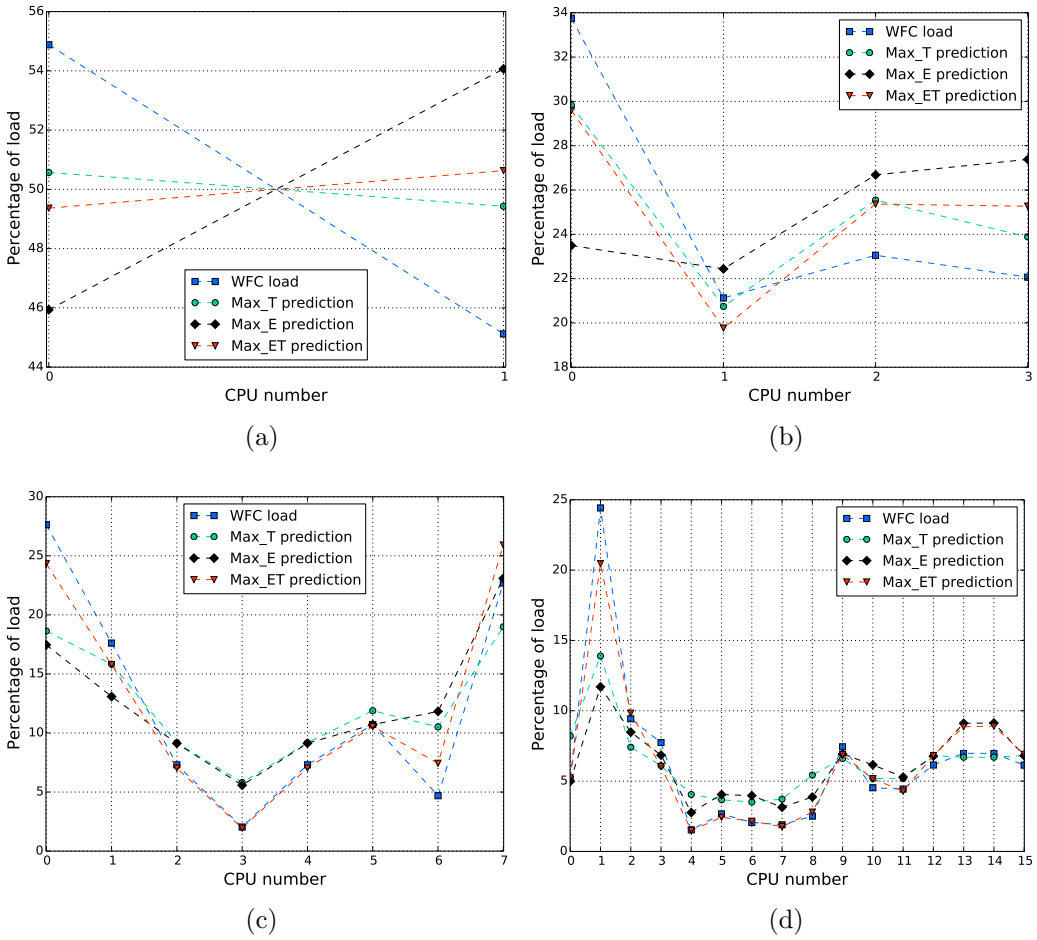


Figure B.1: A comparison between pWFC CPUs percentage load and the predicted percentage load using the preliminary ray tracing used in density prediction for test case 1 with a source located at (1.0,4.5,4.2) km using (a) 2 CPUs, (b) 4 CPUs, (c) 8 CPUs, and (c) 16 CPUs.

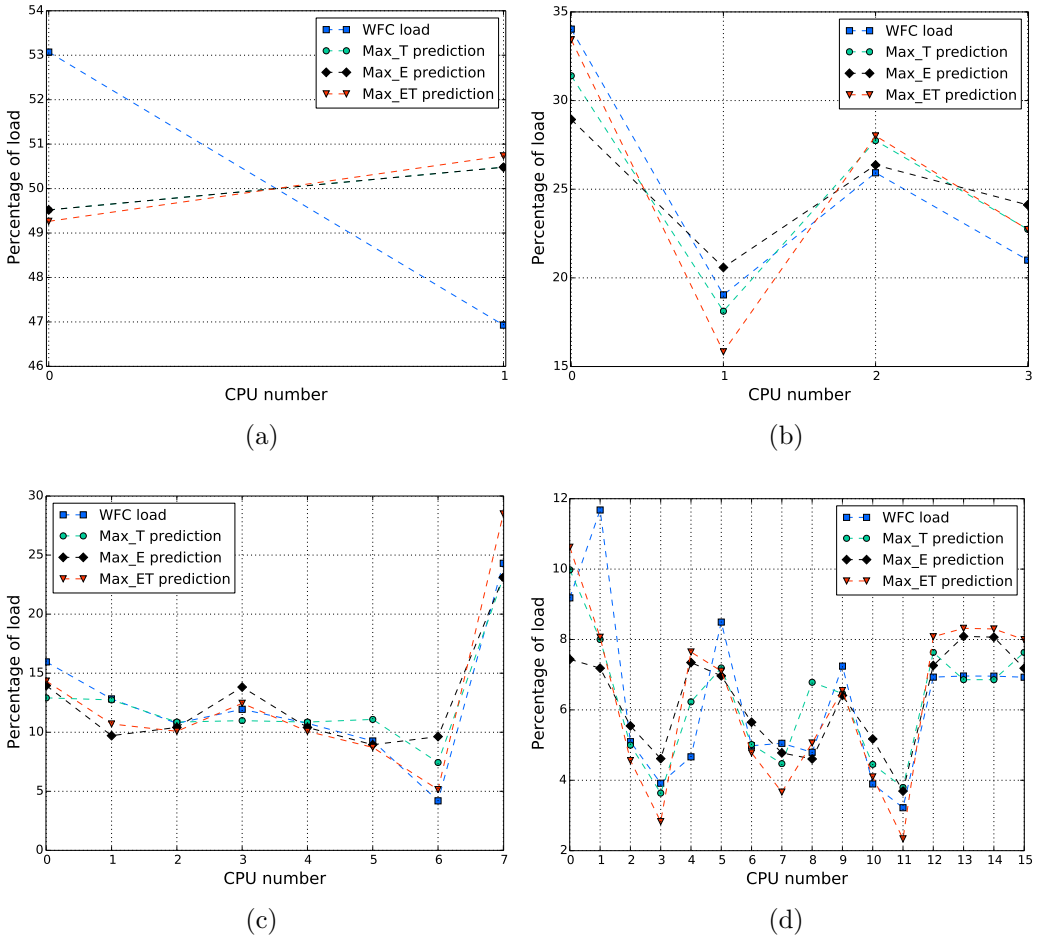


Figure B.2: A comparison between pWFC CPUs percentage load and the predicted percentage load using the preliminary ray tracing used in density prediction for test case 2 with a source located at (4.0,4.5,4.2) km using (a) 2 CPUs, (b) 4 CPUs, (c) 8 CPUs, and (c) 16 CPUs.

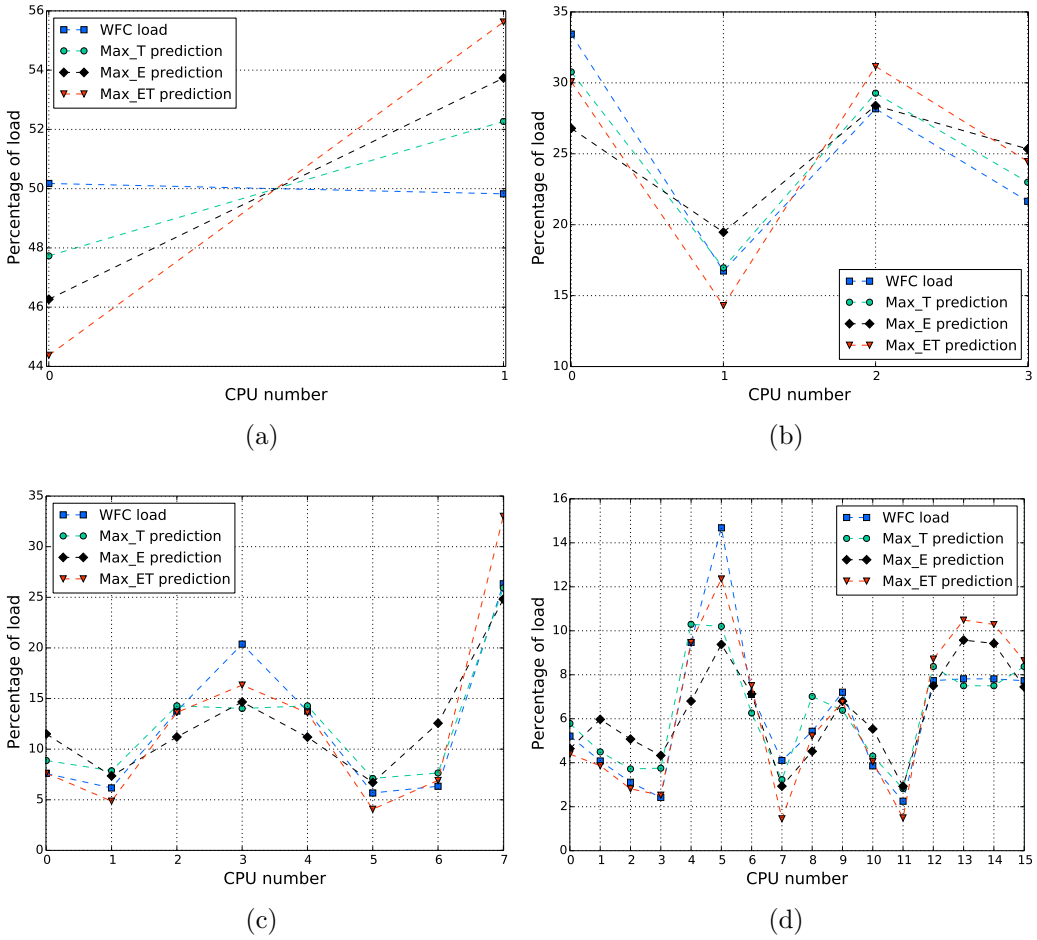
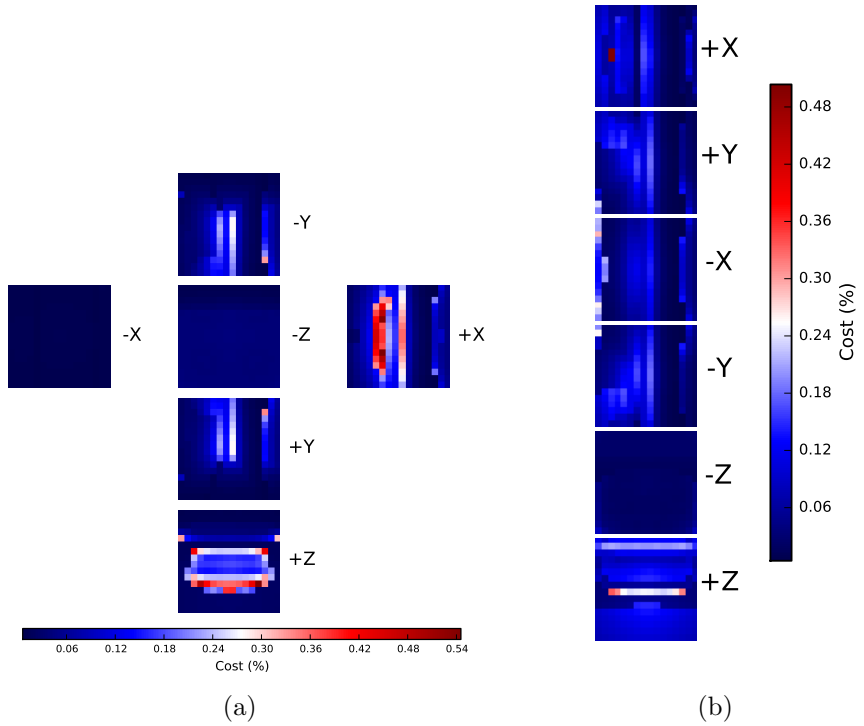
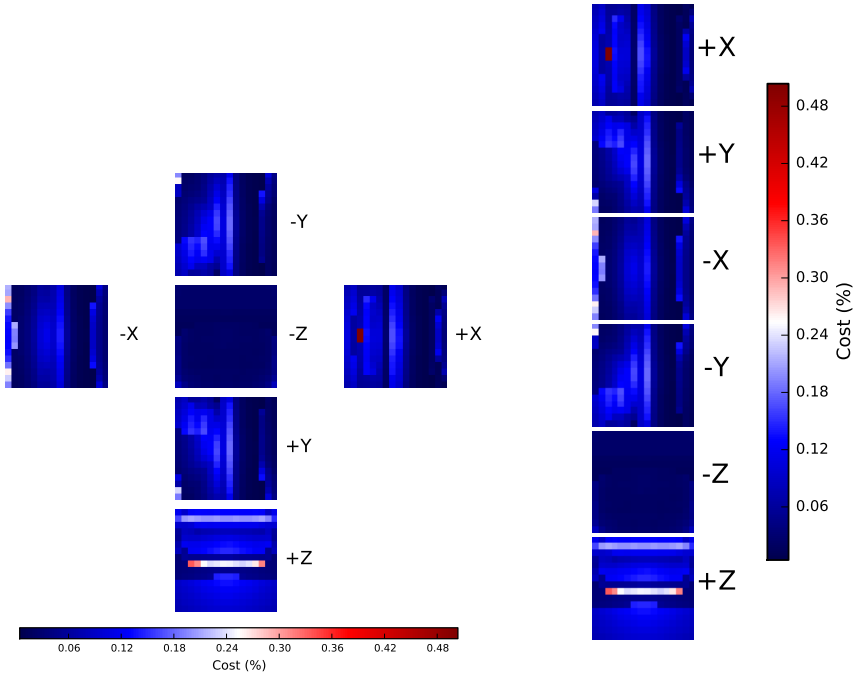


Figure B.3: A comparison between pWFC CPUs percentage load and the predicted percentage load using the preliminary ray tracing used in density prediction for test case 3 with a source located at (7.0,4.5,4.2) km using (a) 2 CPUs, (b) 4 CPUs, (c) 8 CPUs, and (c) 16 CPUs.



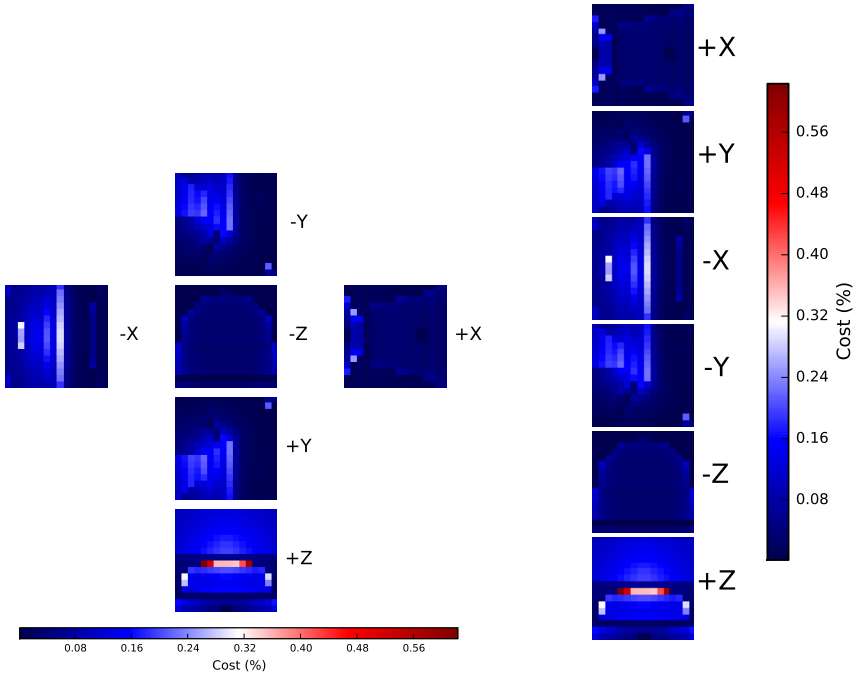
(a) (b)

Figure B.4: Initial wavefront mesh cost distribution between cells using Max_ET approach to prediction for test case 1 with a source located at (1.0,4.5,4.2) km. Each focal cube face represents the direction of the ray tube. Red indicates high potential future load and blue indicates low potential future load. Total cost of each wavefront mesh is 100%. (a) Mesh faces represented using a cube layout, (b) Mesh faces represented using the software implementation layout.



(a) (b)

Figure B.5: Initial wavefront mesh cost distribution between cells using Max_ET approach to prediction for test case 2 with a source located at (4.0,4.5,4.2) km. Each focal cube face represents the direction of the ray tube. Red indicates high potential future load and blue indicates low potential future load. Total cost of each wavefront mesh is 100%. (a) Mesh faces represented using a cube layout, (b) Mesh faces represented using the software implementation layout.



(a) (b)

Figure B.6: Initial wavefront mesh cost distribution between cells using Max_ET approach to prediction for test case 3 with a source located at (7.0,4.5,4.2) km. Each focal cube face represents the direction of the ray tube. Red indicates high potential future load and blue indicates low potential future load. Total cost of each wavefront mesh is 100%. (a) Mesh faces represented using a cube layout, (b) Mesh faces represented using the software implementation layout.

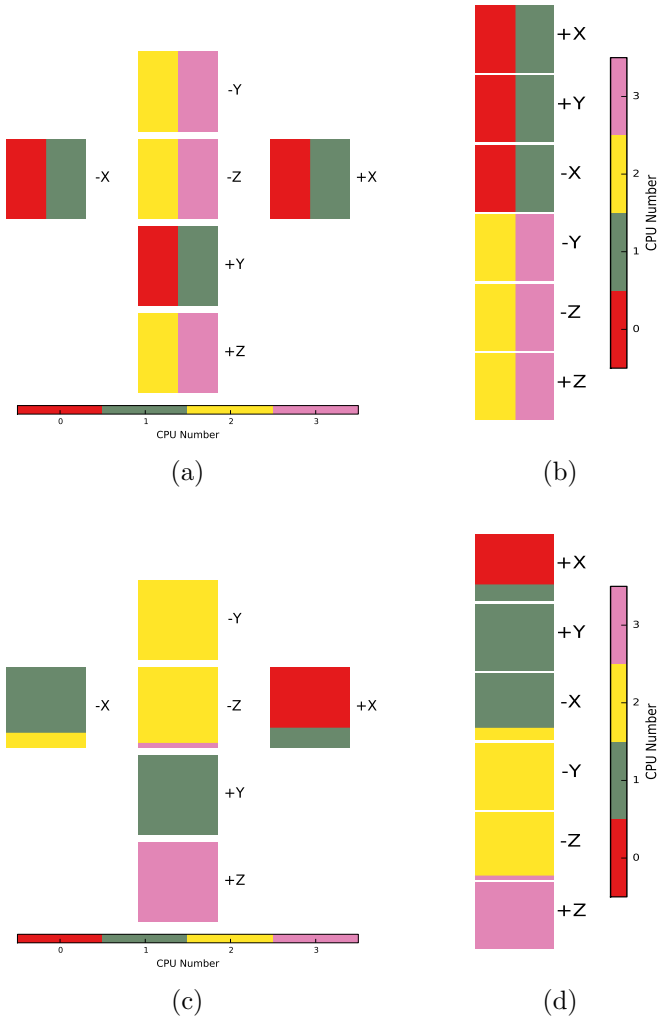


Figure B.7: Initial wavefront mesh partitioning for test case 1 with a source located at (1.0,4.5,4.2) km using 4 CPUs. Each focal cube face represents the direction of the ray tube. (a) Original pWFC implementation using uniform partitioning with mesh faces represented using a cube layout. (b) Original pWFC implementation using uniform partitioning with mesh faces represented using using the software implementation layout. (c) New pWFC implementation using non-uniform partitioning with mesh faces represented using a cube layout. (d) New pWFC implementation using non-uniform partitioning with mesh faces represented using using the software implementation layout.

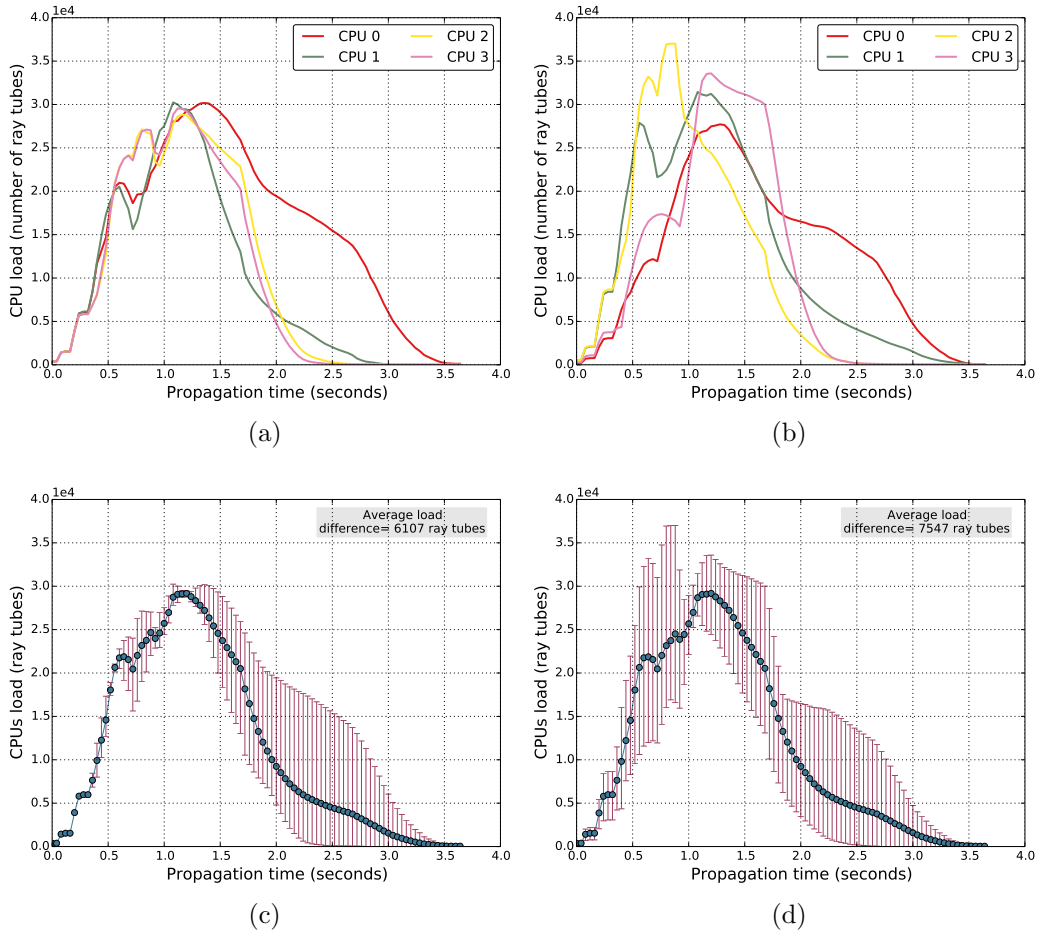


Figure B.8: CPUs load during wavefront propagation for test case 1 with a source located at (1.0,4.5,4.2) km using 4 CPUs.(a) Load profile of original pWFC implementation using uniform partitioning. (b) Load profile of our pWFC implementation using non-uniform partitioning. (c) Average load difference of original pWFC implementation using uniform partitioning. (d) Average load difference of our pWFC implementation using non-uniform partitioning.

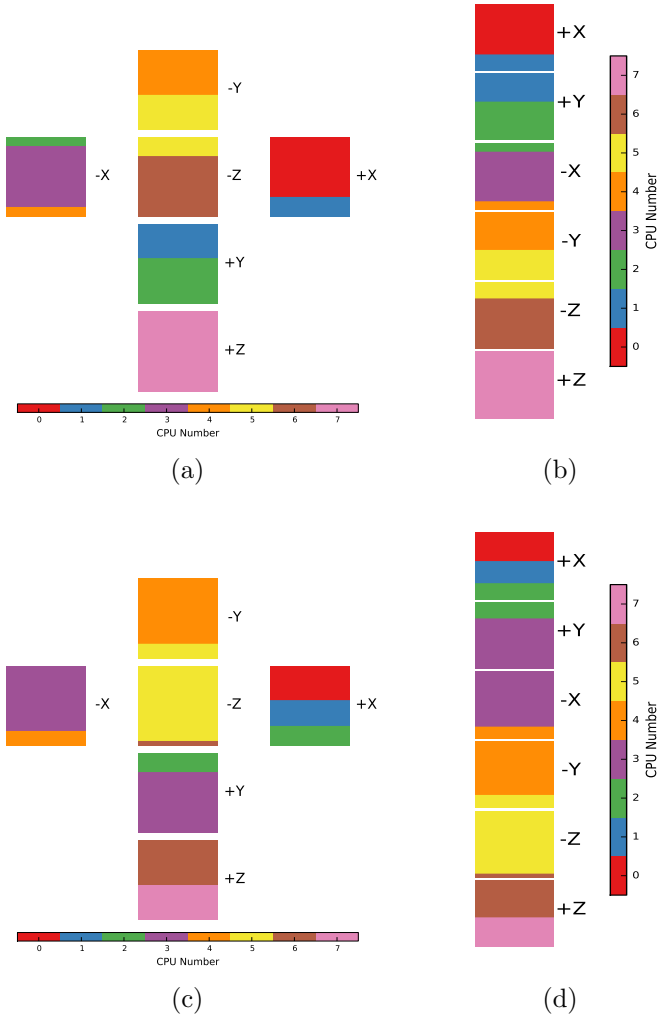


Figure B.9: Initial wavefront mesh partitioning for test case 1 with a source located at (1.0,4.5,4.2) km using 8 CPUs. Each focal cube face represents the direction of the ray tube. (a) Original pWFC implementation using uniform partitioning with mesh faces represented using a cube layout. (b) Original pWFC implementation using uniform partitioning with mesh faces represented using using the software implementation layout. (c) Our pWFC implementation using non-uniform partitioning with mesh faces represented using a cube layout. (d) Our pWFC implementation using non-uniform partitioning with mesh faces represented using using the software implementation layout.

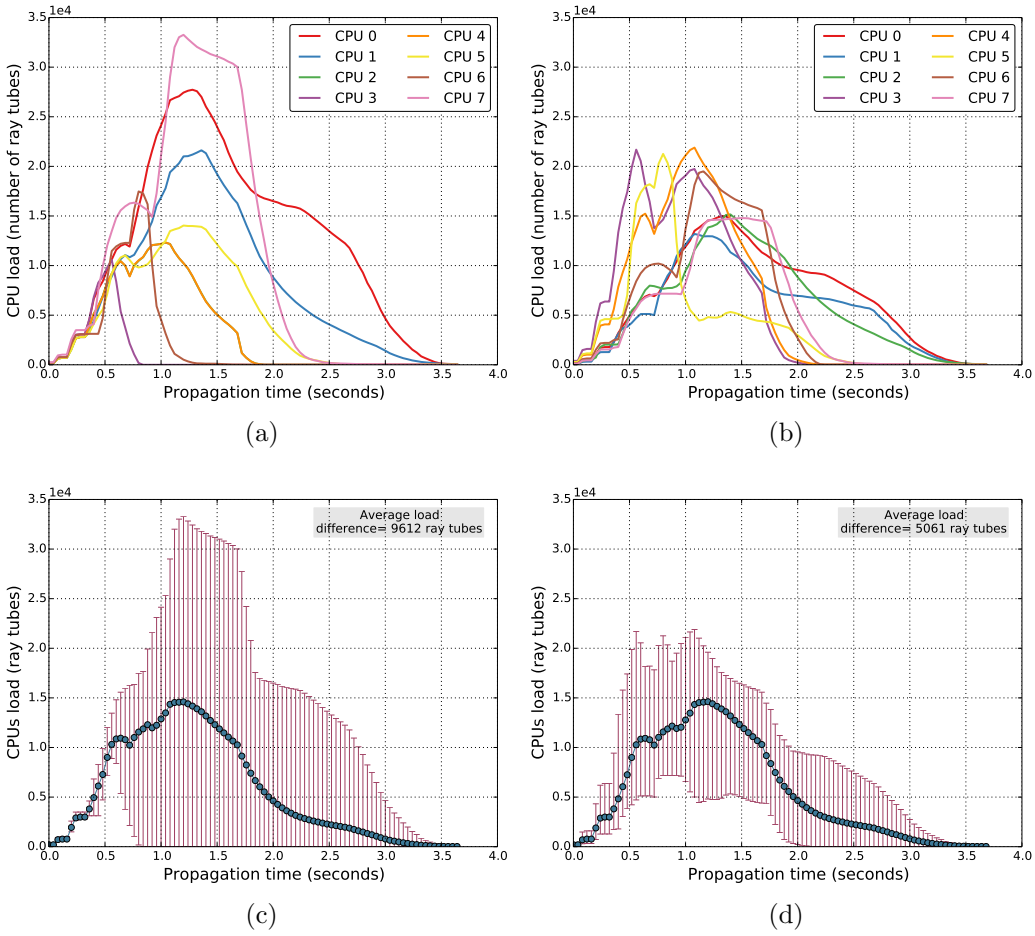


Figure B.10: CPUs load during wavefront propagation for test case 1 with a source located at (1.0,4.5,4.2) km using 8 CPUs. (a) Load profile of original pWFC implementation using uniform partitioning. (b) Load profile of our pWFC implementation using non-uniform partitioning. (c) Average load difference of original pWFC implementation using uniform partitioning. (d) Average load difference of our pWFC implementation using non-uniform partitioning.

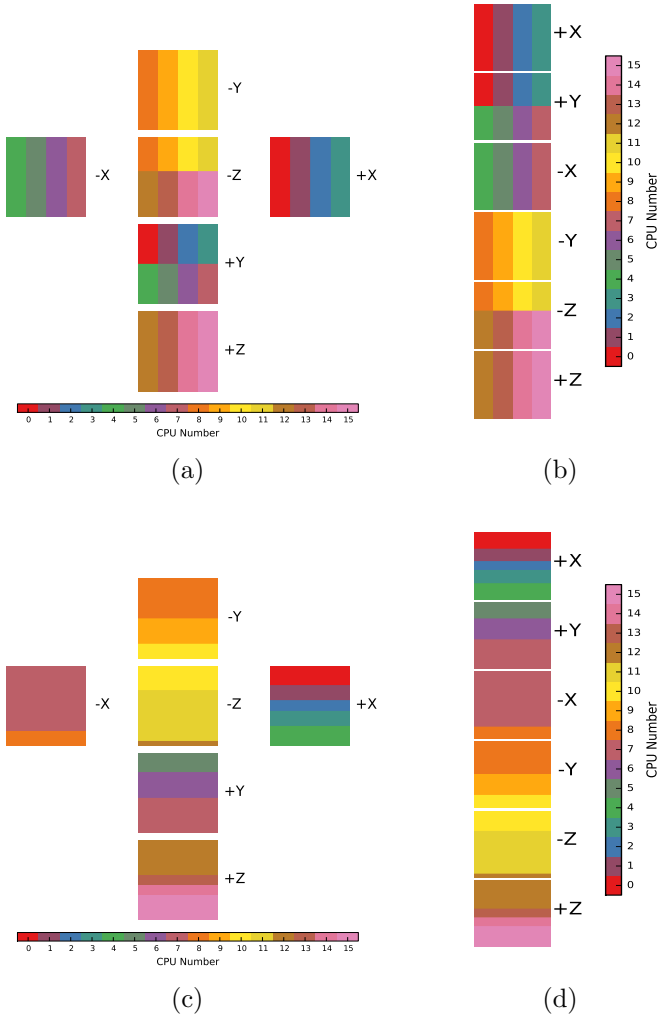


Figure B.11: Initial wavefront mesh partitioning for test case 1 with a source located at (1.0,4.5,4.2) km using 16 CPUs. Each focal cube face represents the direction of the ray tube. (a) Original pWFC implementation using uniform partitioning with mesh faces represented using a cube layout. (b) Original pWFC implementation using uniform partitioning with mesh faces represented using using the software implementation layout. (c) Our pWFC implementation using non-uniform partitioning with mesh faces represented using a cube layout. (d) Our pWFC implementation using non-uniform partitioning with mesh faces represented using using the software implementation layout.

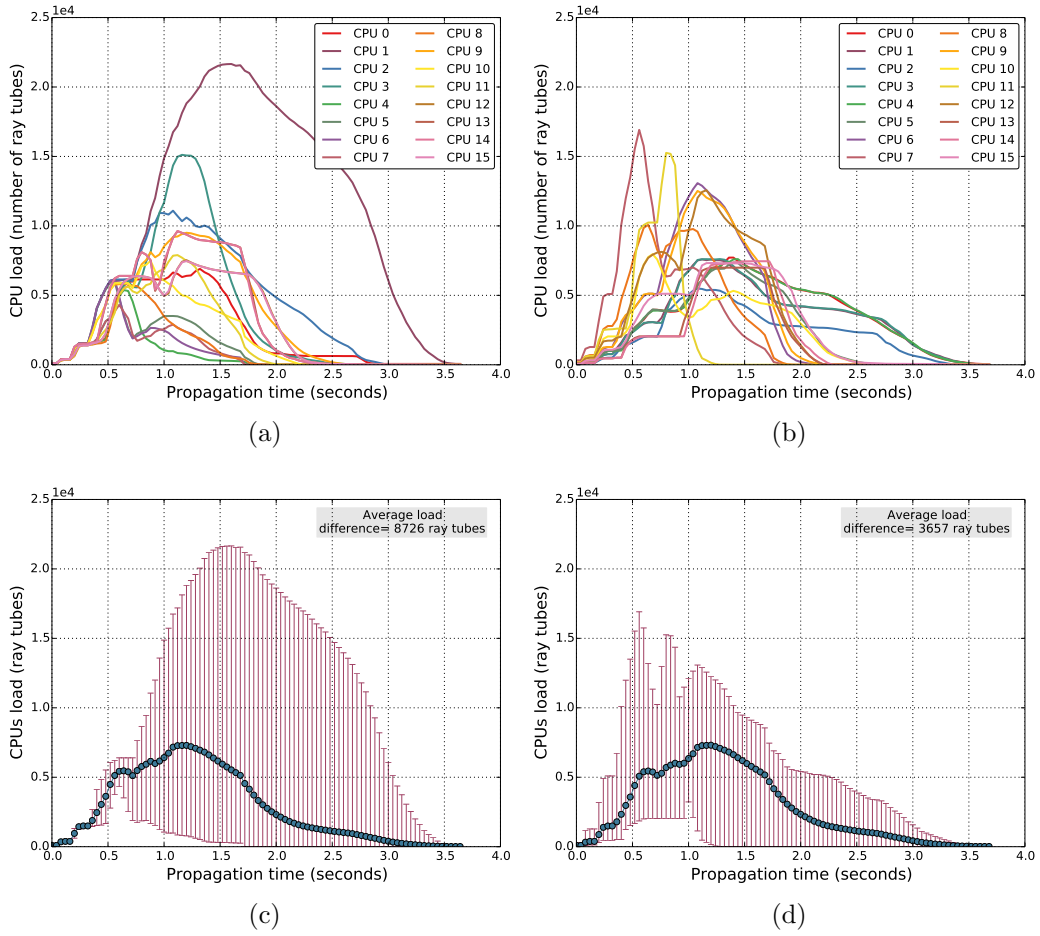


Figure B.12: CPUs load during wavefront propagation for test case 1 with a source located at (1.0,4.5,4.2) km using 16 CPUs. (a) Load profile of original pWFC implementation using uniform partitioning. (b) Load profile of our pWFC implementation using non-uniform partitioning. (c) Average load difference of original pWFC implementation using uniform partitioning. (d) Average load difference of our pWFC implementation using non-uniform partitioning.

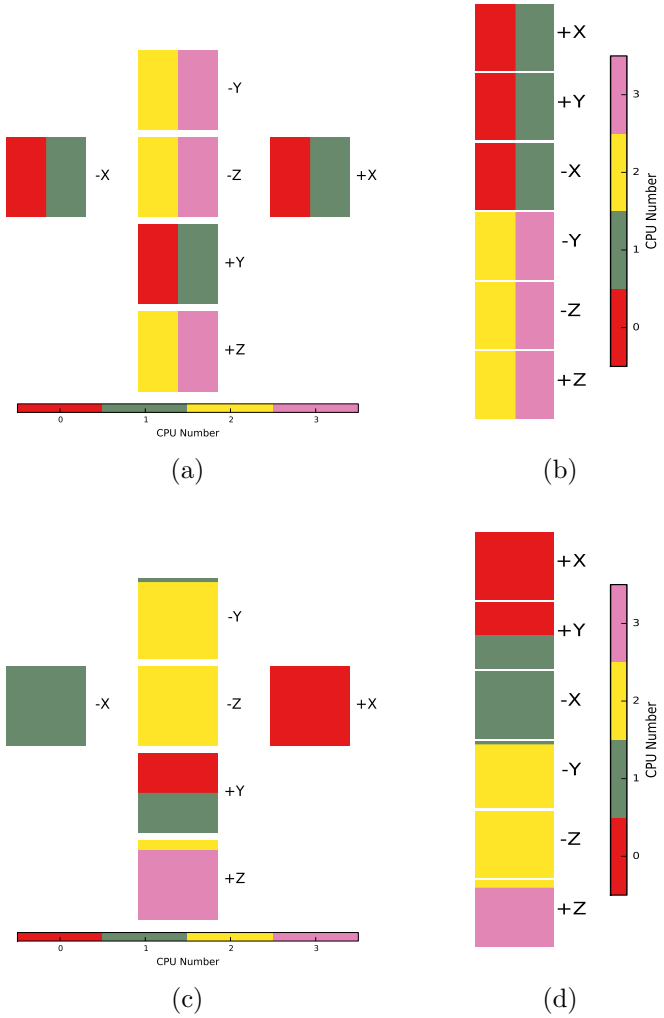


Figure B.13: Initial wavefront mesh partitioning for test case 2 with a source located at (4.0,4.5,4.2) km using 4 CPUs. Each focal cube face represents the direction of the ray tube. (a) Original pWFC implementation using uniform partitioning with mesh faces represented using a cube layout. (b) Original pWFC implementation using uniform partitioning with mesh faces represented using using the software implementation layout. (c) Our pWFC implementation using non-uniform partitioning with mesh faces represented using a cube layout. (d) Our pWFC implementation using non-uniform partitioning with mesh faces represented using using the software implementation layout.

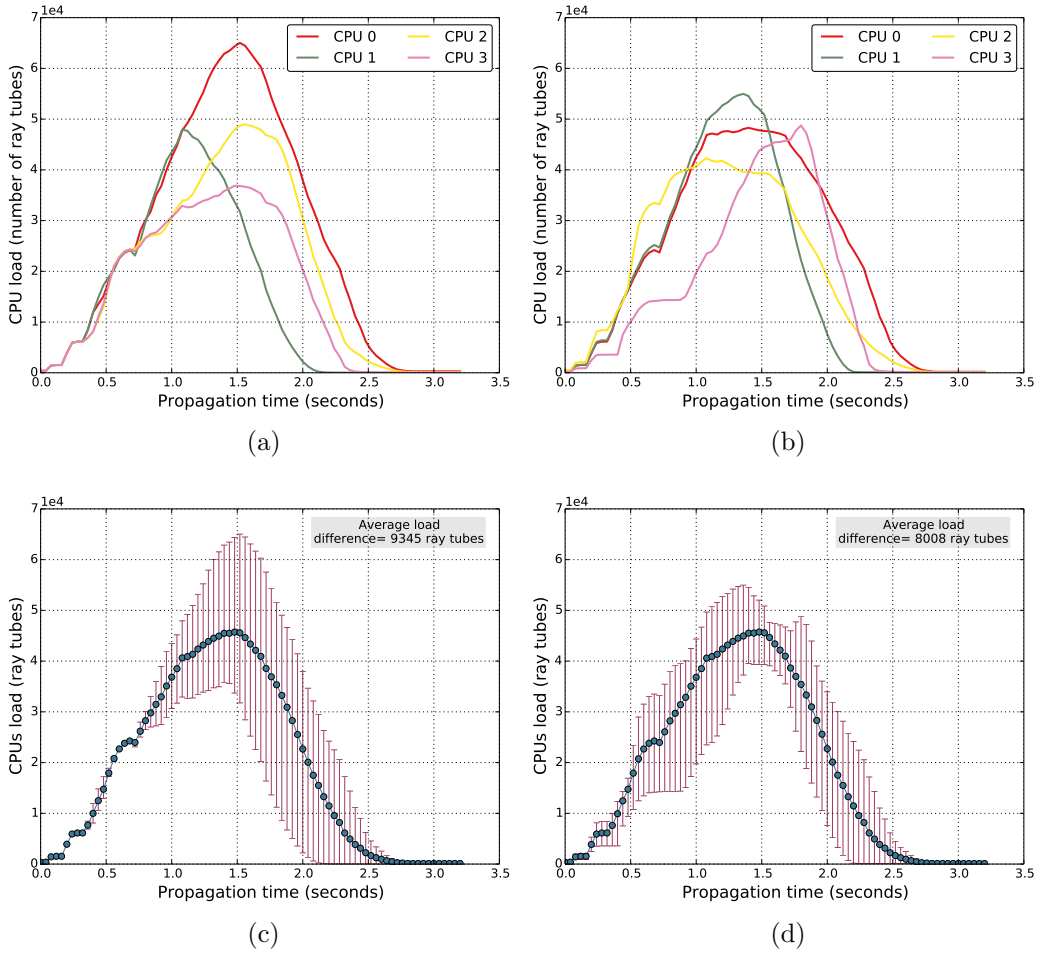


Figure B.14: CPUs load during wavefront propagation for test case 2 with a source located at (4.0,4.5,4.2) km using 4 CPUs. (a) Load profile of original pWFC implementation using uniform partitioning. (b) Load profile of our pWFC implementation using non-uniform partitioning. (c) Average load difference of original pWFC implementation using uniform partitioning. (d) Average load difference of our pWFC implementation using non-uniform partitioning.

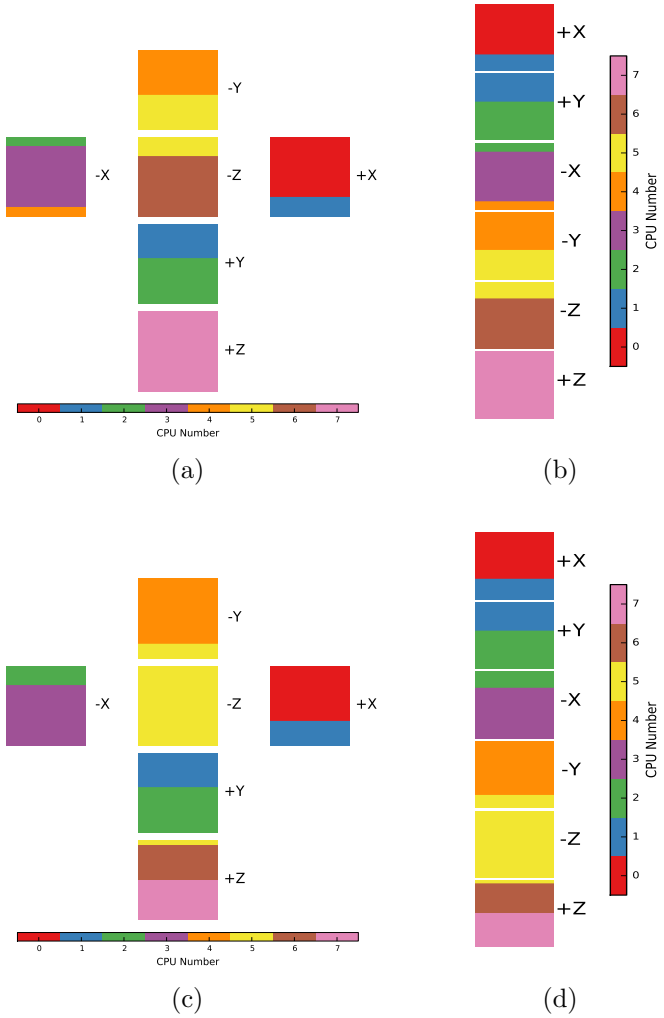


Figure B.15: Initial wavefront mesh partitioning for test case 2 with a source located at (4.0,4.5,4.2) km using 8 CPUs. Each focal cube face represents the direction of the ray tube. (a) Original pWFC implementation using uniform partitioning with mesh faces represented using a cube layout. (b) Original pWFC implementation using uniform partitioning with mesh faces represented using using the software implementation layout. (c) Our pWFC implementation using non-uniform partitioning with mesh faces represented using a cube layout. (d) Our pWFC implementation using non-uniform partitioning with mesh faces represented using using the software implementation layout.

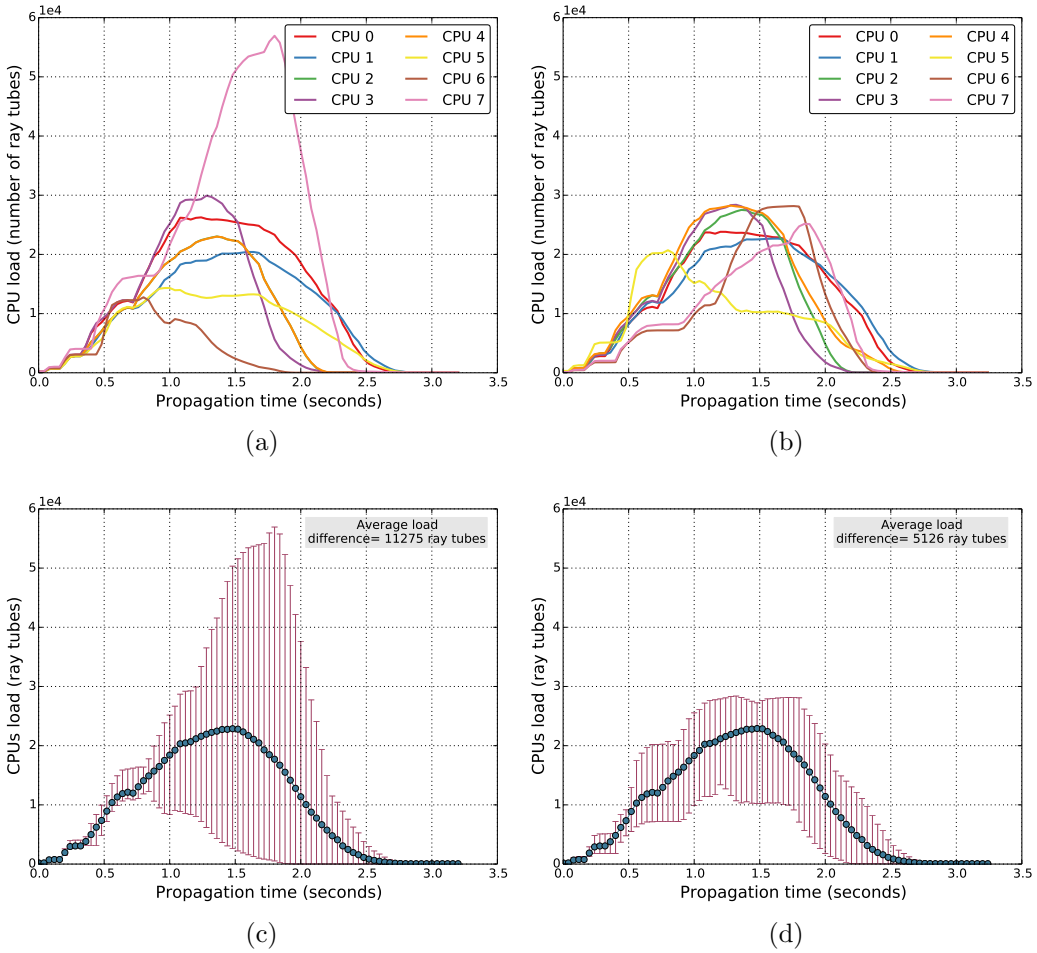


Figure B.16: CPUs load during wavefront propagation for test case 2 with a source located at (4.0,4.5,4.2) km using 8 CPUs. (a) Load profile of original pWFC implementation using uniform partitioning. (b) Load profile of our pWFC implementation using non-uniform partitioning. (c) Average load difference of original pWFC implementation using uniform partitioning. (d) Average load difference of our pWFC implementation using non-uniform partitioning.

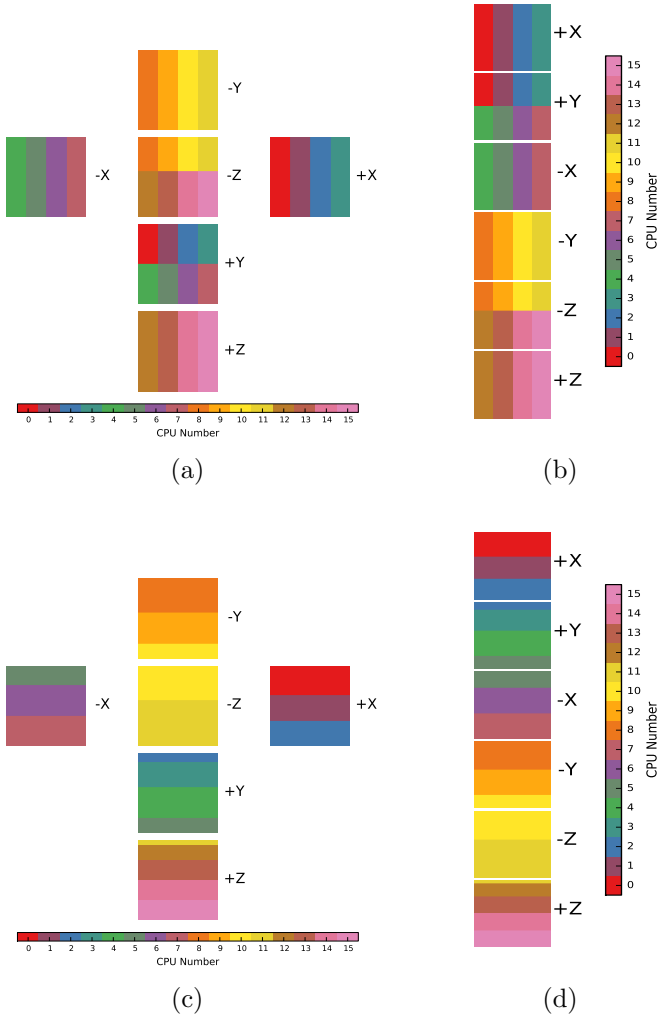


Figure B.17: Initial wavefront mesh partitioning for test case 2 with a source located at (4.0,4.5,4.2) km using 16 CPUs. Each focal cube face represents the direction of the ray tube. (a) Original pWFC implementation using uniform partitioning with mesh faces represented using a cube layout. (b) Original pWFC implementation using uniform partitioning with mesh faces represented using using the software implementation layout. (c) Our pWFC implementation using non-uniform partitioning with mesh faces represented using a cube layout. (d) Our pWFC implementation using non-uniform partitioning with mesh faces represented using using the software implementation layout.

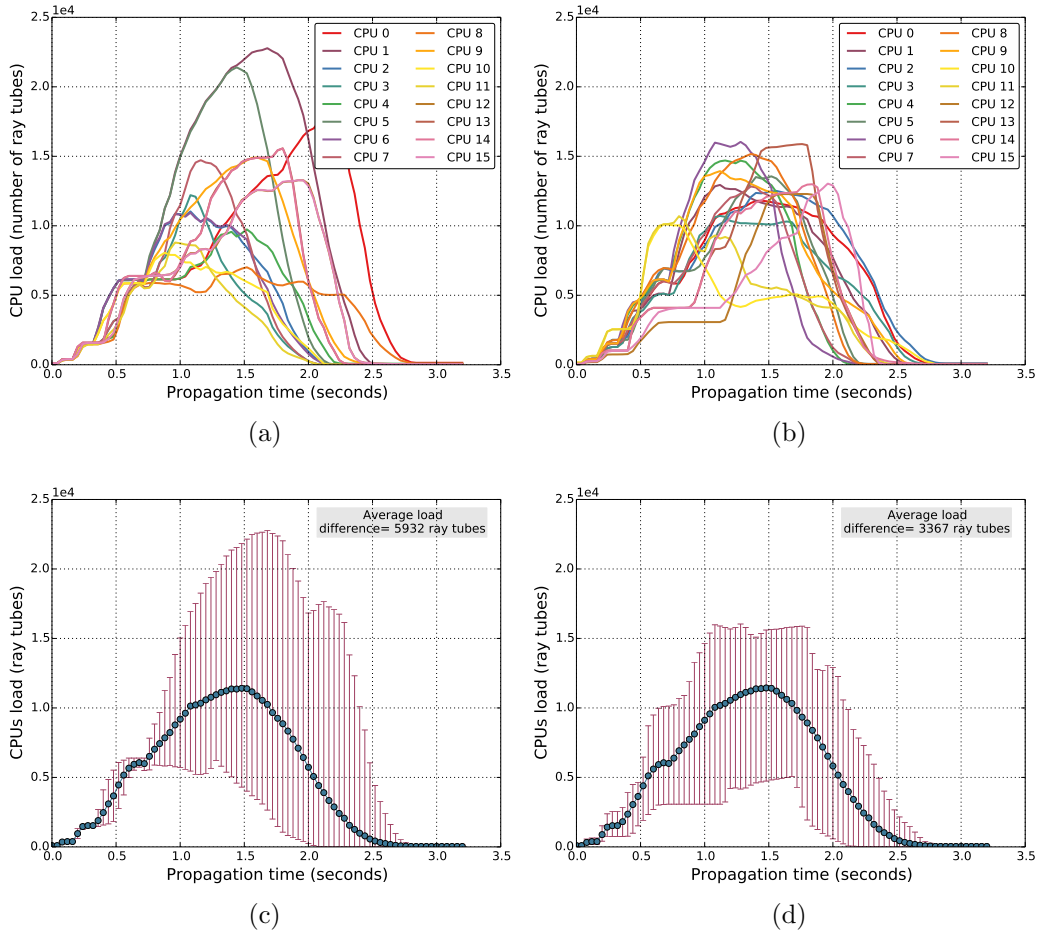


Figure B.18: CPUs load during wavefront propagation for test case 2 with a source located at (4.0,4.5,4.2) km using 16 CPUs. (a) Load profile of original pWFC implementation using uniform partitioning. (b) Load profile of our pWFC implementation using non-uniform partitioning. (c) Average load difference of original pWFC implementation using uniform partitioning. (d) Average load difference of our pWFC implementation using non-uniform partitioning.

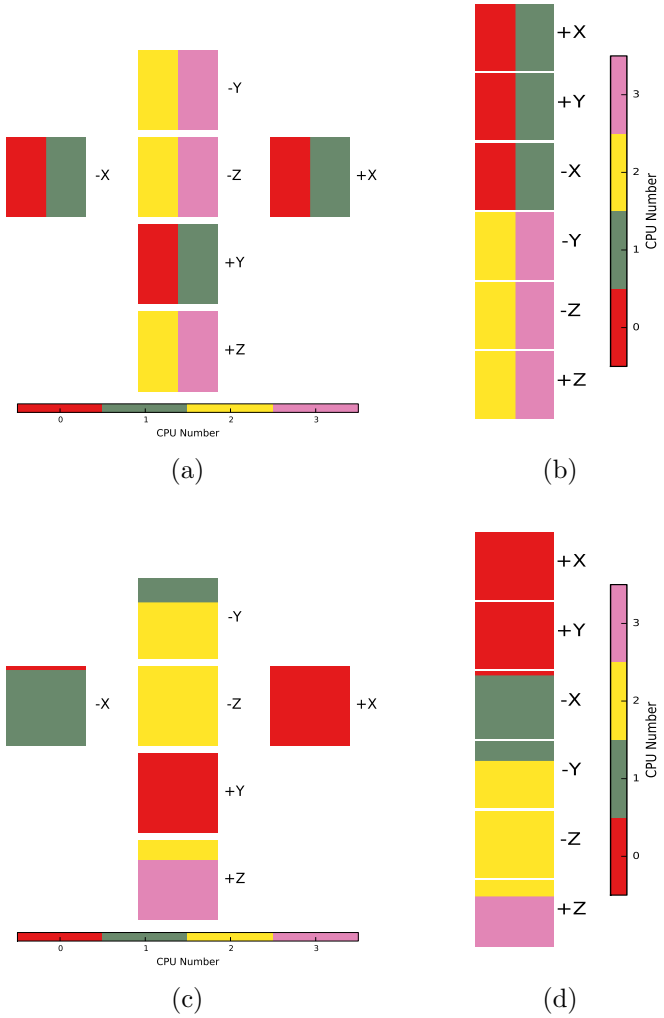


Figure B.19: Initial wavefront mesh partitioning for test case 3 with a source located at (7.0,4.5,4.2) km using 4 CPUs. Each focal cube face represents the direction of the ray tube. (a) Original pWFC implementation using uniform partitioning with mesh faces represented using a cube layout. (b) Original pWFC implementation using uniform partitioning with mesh faces represented using using the software implementation layout. (c) Our pWFC implementation using non-uniform partitioning with mesh faces represented using a cube layout. (d) Our pWFC implementation using non-uniform partitioning with mesh faces represented using using the software implementation layout.

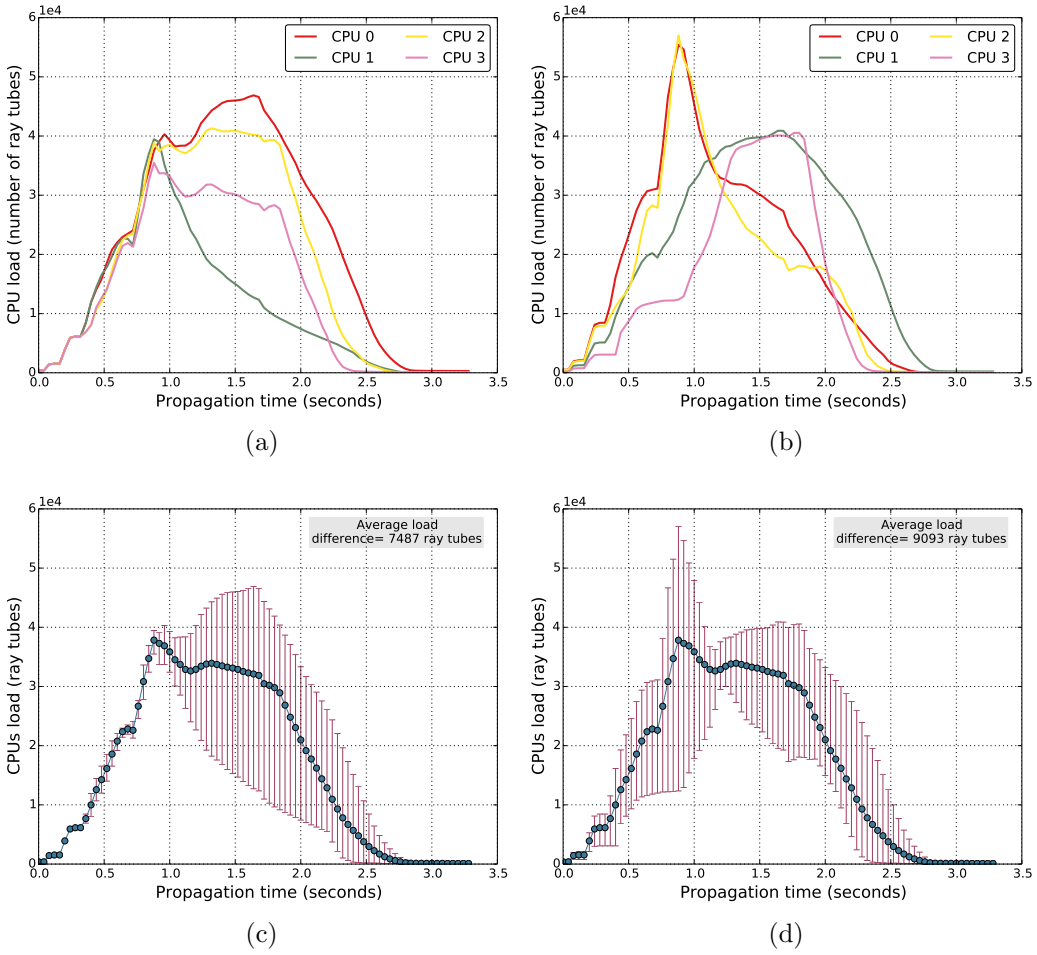


Figure B.20: CPUs load during wavefront propagation for test case 3 with a source located at (7.0,4.5,4.2) km using 4 CPUs. (a) Load profile of original pWFC implementation using uniform partitioning. (b) Load profile of our pWFC implementation using non-uniform partitioning. (c) Average load difference of original pWFC implementation using uniform partitioning. (d) Average load difference of our pWFC implementation using non-uniform partitioning.

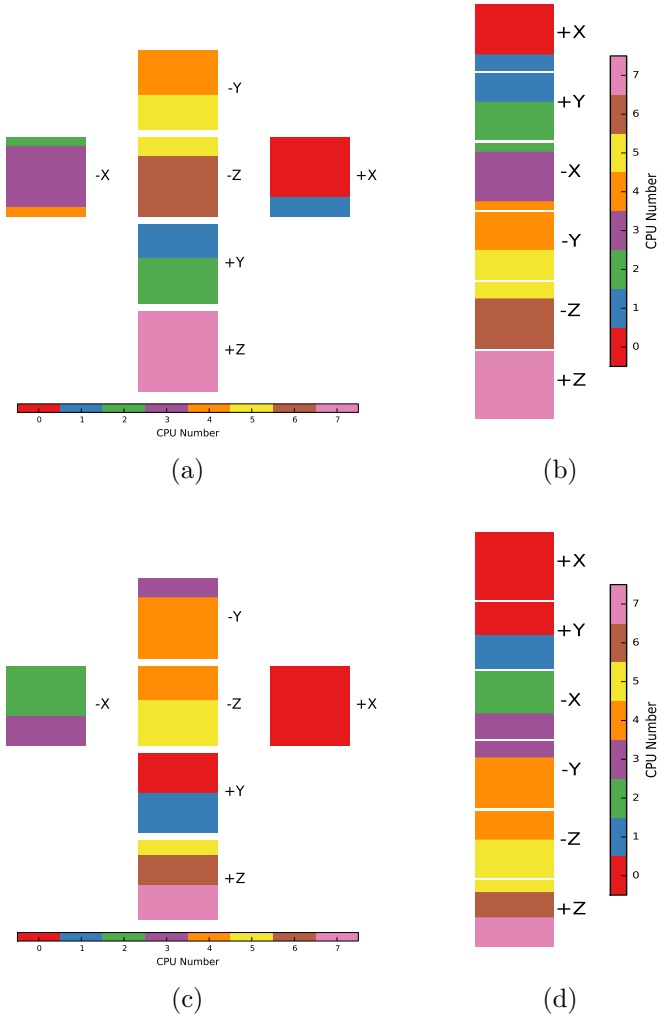


Figure B.21: Initial wavefront mesh partitioning for test case 3 with a source located at (7.0,4.5,4.2) km using 8 CPUs. Each focal cube face represents the direction of the ray tube. (a) Original pWFC implementation using uniform partitioning with mesh faces represented using a cube layout. (b) Original pWFC implementation using uniform partitioning with mesh faces represented using using the software implementation layout. (c) Our pWFC implementation using non-uniform partitioning with mesh faces represented using a cube layout. (d) Our pWFC implementation using non-uniform partitioning with mesh faces represented using using the software implementation layout.

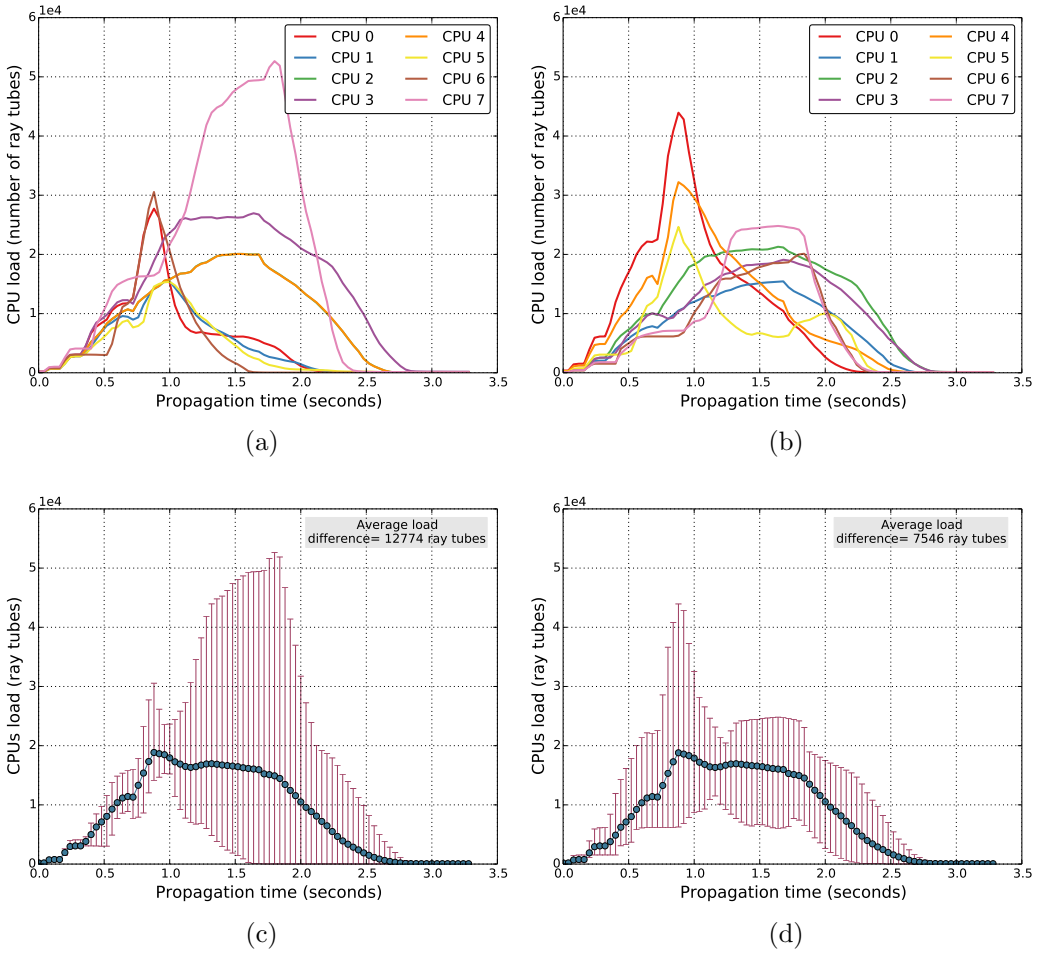


Figure B.22: CPUs load during wavefront propagation for test case 3 with a source located at (7.0,4.5,4.2) km using 8 CPUs. (a) Load profile of original pWFC implementation using uniform partitioning. (b) Load profile of our pWFC implementation using non-uniform partitioning. (c) Average load difference of original pWFC implementation using uniform partitioning. (d) Average load difference of our pWFC implementation using non-uniform partitioning.

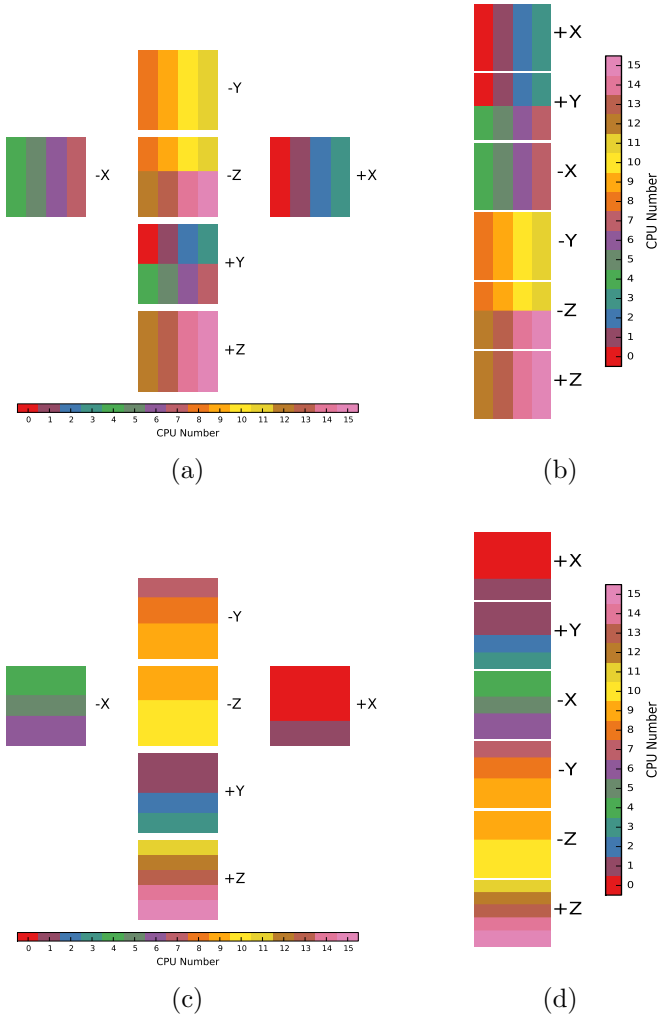


Figure B.23: Initial wavefront mesh partitioning for test case 3 with a source located at (7.0,4.5,4.2) km using 16 CPUs. Each focal cube face represents the direction of the ray tube. (a) Original pWFC implementation using uniform partitioning with mesh faces represented using a cube layout. (b) Original pWFC implementation using uniform partitioning with mesh faces represented using using the software implementation layout. (c) Our pWFC implementation using non-uniform partitioning with mesh faces represented using a cube layout. (d) Our pWFC implementation using non-uniform partitioning with mesh faces represented using using the software implementation layout.

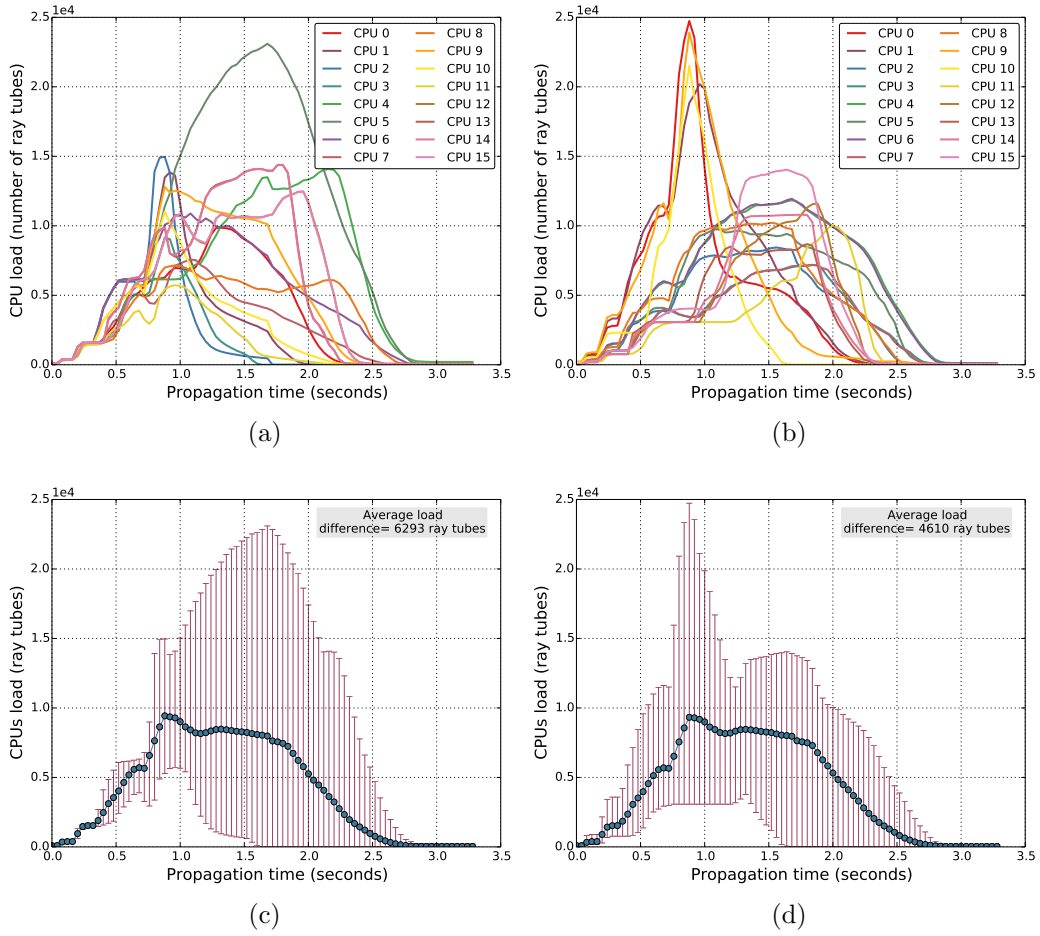


Figure B.24: CPUs load during wavefront propagation for test case 3 with a source located at (7.0,4.5,4.2) km using 16 CPUs. (a) Load profile of original pWFC implementation using uniform partitioning. (b) Load profile of our pWFC implementation using non-uniform partitioning. (c) Average load difference of original pWFC implementation using uniform partitioning. (d) Average load difference of our pWFC implementation using non-uniform partitioning.

APPENDIX C

SUPPLEMENTARY MEDIA FILES

Supplementary media for this thesis show the WFC method wave simulations and 3D preliminary ray tracing (without interpolation) for each executed test case in the targeted earth model (salt dome). File names and descriptions is provided as follows:

l1_cost_1.m4v and l1_cost_20.m4v: Shows the preliminary ray tracing for test case 1 with a source located at (1.0,4.5,4.2) km.

l2_cost_1.m4v and l2_cost_20.m4v: Shows the preliminary ray tracing for test case 2 with a source located at (4.0,4.5,4.2) km.

l3_cost_1.m4v and l3_cost_20.m4v: Shows the preliminary ray tracing for test case 3 with a source located at (7.0,4.5,4.2) km.

l1_wave_1.m4v and l1_wave_20.m4v: Shows the WFC method wavefront simulation for test case 1 with a source located at (1.0,4.5,4.2) km.

l2_wave_1.m4v and l2_wave_20.m4v: Shows the WFC method wavefront simulation for test case 2 with a source located at (4.0,4.5,4.2) km.

l3_wave_1.m4v and l3_wave_20.m4v: Shows the WFC method wavefront simulation for test case 3 with a source located at (7.0,4.5,4.2) km.